5-2017

Inventory Optimization Using a SimPy Simulation Model

Lauren Holden
East Tennessee State University

Inventory Optimization Using a SimPy Simulation Model

A thesis

presented to

the faculty of the Department of Mathematics

East Tennessee State University

In partial fulfillment

of the requirements for the degree

Master of Science in Mathematical Sciences

---

by

Lauren Holden

May 2017

Jeff Knisley, Ph.D., Chair

Baptiste Lebreton, Ph.D.

Michele Joyner, Ph.D.

Robert Price, Ph.D.

Keywords: inventory, optimization, safety stock, guaranteed service time, SimPy

ABSTRACT

Inventory Optimization Using a SimPy Simulation Model

by

Lauren Holden

Existing multi-echelon inventory optimization models and formulas were studied to get an understanding of how safety stock levels are determined. Because of the restrictive distribution assumptions of the existing safety stock formula, which are not necessarily realistic in practice, a method to analyze the performance of this formula in a more realistic setting was desired. A SimPy simulation model was designed and implemented for a simple two-stage supply chain as a way to test the performance of the safety stock formula. This implementation produced results which led to the conclusion that the safety stock formula tends to underestimate the level of safety stock needed to provide a certain service level when predicted standard deviation of demand is underestimated and the assumptions of normally distributed demand and normally distributed lead times are not fulfilled.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Jeff Knisley, for all of his input and guidance throughout this process. I would also like to thank Dr. Baptiste Lebreton for taking the time to work with me on this project and provide guidance and support along the way. Finally, I would like to thank the members of my advisory committee, Dr. Robert Price and Dr. Michele Joyner, for their input and suggestions, both on this project and throughout my time as a student.

TABLE OF CONTENTS

ABSTRACT 2

ACKNOWLEDGMENTS 4

LIST OF TABLES 7

LIST OF FIGURES 8

1 INTRODUCTION AND BACKGROUND 10

1.1 Supply Chain Definition and Connection to Inventory Optimization 10

1.2 Inventory Definition and Types 12

1.3 Inventory Optimization 17

1.4 Simulating the Supply Chain Using SimPy 20

2 MATHEMATICAL FORMULAS, DISTRIBUTIONS, AND MODELS 22

2.1 Safety Stock Formula 22

2.2 Related Probability Distributions 23

2.3 Graves and Willems Model 25

2.3.1 Single-Stage Model 27

2.3.2 Multi-Stage Model 30

2.4 Guaranteed Service Time 31

3 THE SIMPY SIMULATION MODEL 33

3.1 Introduction to Simulation Software and Model 33

3.2 Components of the Model 34

3.2.1 External Components 35

3.2.2 Internal Components 35

3.3 Implementation of BU Node 36
3.4 Implementation of DR Node 37
3.5 Monitoring Within the Simulation 38
3.5.1 External Monitoring 39
3.5.2 Internal Monitoring 39
3.6 Running the Simulation 40
3.7 Production of Simulation Results 41
4 RESULTS AND CONCLUSIONS 42
4.1 Model Validation 43
4.2 Analysis of Safety Stock Equation 47
4.3 Model Variations to Consider 61
4.4 Concluding Remarks 61
BIBLIOGRAPHY 64
APPENDICES 66
A SimPy Model 66
B Shapiro-Wilk Test Results for Simulated Demand 73
VITA 74

LIST OF TABLES

1 Description of model entities related to BU node by class, including name, type, function, and all related parameters 37

2 Description of DR model entities by class, including name, type, function, and all related parameters 38

3 Mean and standard deviation of predicted BU safety stock requirements across five simulation runs by cycle service level 60

4 Mean and standard deviation of predicted DR safety stock requirements across five simulation runs by cycle service level 60

LIST OF FIGURES

1 Illustration of a simple supply chain 11

2 Illustration of a more complex supply chain 11

3 Inventory profile illustrating cycle inventory under the assumption of steady demand 14

4 Inventory profile illustrating safety inventory as a buffer against supply uncertainty 16

5 Relationship between service level, savings, and costs 19

6 Illustration of supply chain for simulation model 36

7 Plots of simulated BU demand with daily mean of 50 44

8 Plots of simulated DR demand with daily mean of 50 44

9 Plots of required safety stock by cycle service level for BU based on assumed and simulated demand 46

10 Plots of required safety stock by cycle service level for DR based on assumed and simulated demand 47

11 Plots of simulated BU demand with daily mean of $\frac{5}{60}$ 48

12 Plots of simulated DR demand with daily mean of $\frac{1}{2}$ 49

13 Plot of required safety stock by cycle service level for BU for simulation Run 1 50

14 Plot of required safety stock by cycle service level for BU for simulation Run 2 51

15 Plot of required safety stock by cycle service level for BU for simulation Run 3 52

16 Plot of required safety stock by cycle service level for BU for simulation Run 4 53
17 Plot of required safety stock by cycle service level for BU for simulation Run 5 54
18 Plot of required safety stock by cycle service level for DR for simulation Run 1 55
19 Plot of required safety stock by cycle service level for DR for simulation Run 2 56
20 Plot of required safety stock by cycle service level for DR for simulation Run 3 57
21 Plot of required safety stock by cycle service level for DR for simulation Run 4 58
22 Plot of required safety stock by cycle service level for DR for simulation Run 5 59

1 INTRODUCTION AND BACKGROUND

The problem at hand is improving upon the existing methods used by companies to decide the level of safety stock they should hold. In order to do this, a foundation for the problem must be built. Chapter 1 briefly reviews some necessary background material and introduces the problem. Chapter 2 is a discussion of existing formulas and models used in inventory optimization, as well as a brief introduction to SimPy. Chapter 3 explains how SimPy simulations can be used to model supply chains and analyze the validity of the existing formulas in inventory optimization. Chapter 4 is a summary and discussion of the results obtained in Chapter 3.

1.1 Supply Chain Definition and Connection to Inventory Optimization

The definition of a supply chain differs depending on what source is referenced, but they all have the same underlying idea. According to [3], a supply chain is defined as "a network of connected and interdependent organizations mutually and cooperatively working together to control, manage, and improve the flow of materials and information from suppliers to end users." Supply chains can vary in size and complexity, with some having only a few stages while others have several. For example, one supply chain may consist of a supplier of raw materials, a manufacturing plant, and a retailer who sells to consumers, as illustrated in Figure 1. However, another supply chain may consist of several suppliers, several separate manufacturing processes for individual components, a manufacturer that assembles these components into a finished product, several distribution centers, and retailers throughout the country who sell to consumers, as illustrated in Figure 2.

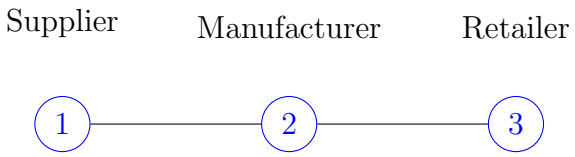


Figure 1: Illustration of a simple supply chain consisting of a supplier of raw materials (node 1), a manufacturer that turns those raw materials into a finished product (node 2), and a retailer who sells the finished product to consumers (node 3).

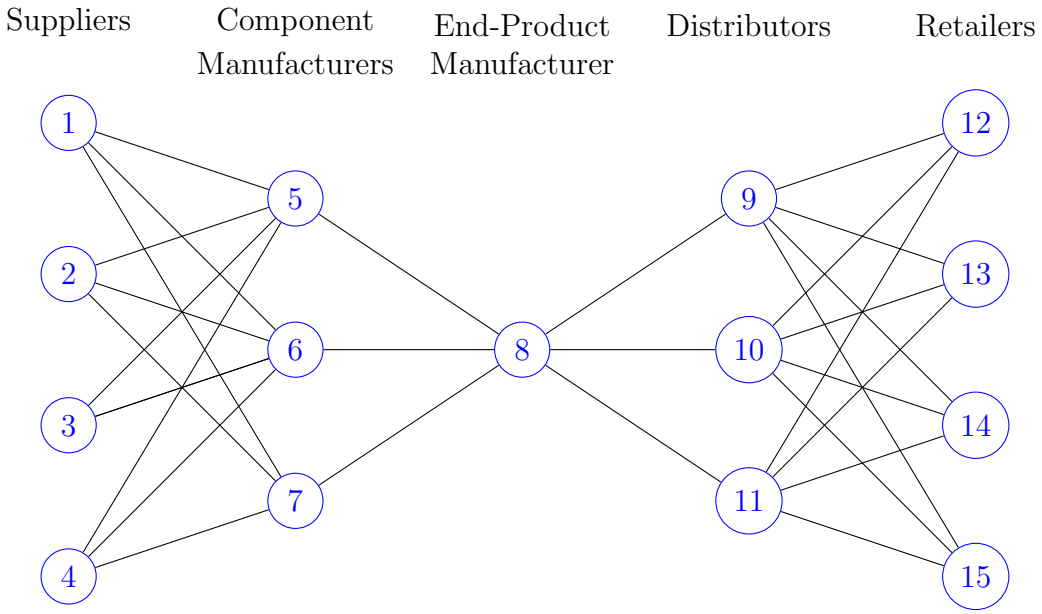


Figure 2: Illustration of a more complex supply chain consisting of suppliers (nodes 1-4), several separate manufacturing processes for individual components (nodes 5-7), a manufacturer that assembles these components into a finished product (node 8), several distribution centers (nodes 9-11), and retailers who sell to consumers (nodes 12-15).

As illustrated in Figures 1 and 2, supply chain makeups differ greatly; however, every product reaches customers through a supply chain of some kind [18]. This means that supply chains are a huge component of inventory optimization, which is the topic of Section 1.3. In fact, inventory, which is discussed in the next section, can potentially be held at each stage along the supply chain. In order to determine the optimal locations to hold inventory and the levels to hold, we will simulate the supply chain using a Python simulation package called SimPy, as discussed in Section 1.4. This will allow us to get a better idea of how the formulas and models perform under the actual conditions of the specific supply chain being analyzed.

1.2 Inventory Definition and Types

Inventory is often thought of as the food sitting on the shelves at a grocery store or the clothes hanging on the racks at a retail store. These are examples of inventory, and they successfully illustrate a common materials management definition of inventory, which states that inventory is "a usable but idle resource having some economic value" [16]. However, understanding what inventory is and why it is needed is much more complicated than these examples imply. At the most basic level, inventory is present because the levels of supply and demand in a supply chain are not equal [2]. However, all inventory is not the same. Inventory can be broken down into several types based on the role the inventory plays in the supply chain. In this paper, we will look at five of these types of inventory: cycle, safety, work in progress, pre-build, and waste.

According to [2], cycle inventory in a supply chain is the average amount of inventory used to satisfy customer demand during a cycle. When demand is constant, for a

lot size of Q, the cycle inventory is $\frac{Q}{2}$. As illustrated in Figure 3 below, when demand is constant and no supply uncertainty is present, the inventory level at the beginning of a cycle is Q, and the inventory level at the end of a cycle is 0. This implies that the average amount of inventory in a cycle is equal to $\frac{Q}{2}$. When demand is varying, the cycle inventory is still proportional to the lot size, but it not necessarily half of it. This type of inventory exists because companies purchase or produce products in lot sizes that exceed customer demand in order to take advantage of economies of scale. Within a certain range, fixed costs are not affected by the quantity of products produced or purchased, so companies order or produce quantities at the high end of this range in order to drive down the per-unit cost of the product [2]. An inventory profile illustrating cycle inventory can be seen in Figure 3, where the following notation is used. For the purposes of Figure 3, we assume demand is constant.

- Q = Lot size: Quantity of product ordered
- A_k = Arrival time in cycle k: Time at which replenishment order arrives

As illustrated in Figure 3, cycle inventory is sufficient to satisfy all demand when demand is constant because companies know exactly what to expect. However, this assumption is not realistic, so we must consider another type of inventory. Safety inventory in a supply chain exists to fulfill customer demand that exceeds what is expected or predicted or to act as a buffer against supply uncertainty [2]. The level of demand for the majority of products is far from constant [7], and although there are tools to predict what the demand will be, there is always some uncertainty. When customers demand more than what is predicted, demand cannot be met and, often-

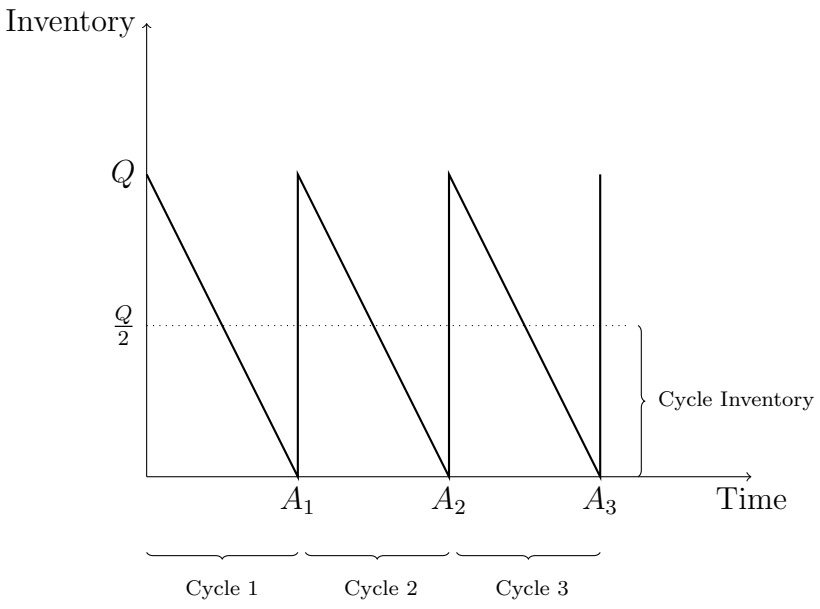


Figure 3: Inventory profile illustrating cycle inventory under the assumption of steady demand: Cycle inventory, the average inventory in a cycle, is equal to $\frac{Q}{2}$ since demand is steady. Under this assumption, demand will be equal to the predicted demand, so the exact amount of cycle inventory needed to satisfy demand in Cycle 1, Cycle 2, and Cycle 3 can be held.

times, the orders are cancelled all together, which is known as a stock out [7]. This leads to unhappy customers and oftentimes a loss in profit and business in the long term. Uncertainty in supply, whether it is caused by a machine breakdown, transportation delay, problem with product quality, or another unforeseen problem, is also a cause for the holding of safety inventory [2]. Regardless of the cause, if an upstream stage fails to supply the desired level of product to the downstream stage(s), demand cannot be met with cycle inventory alone. For these reasons, safety stock is carried

as a buffer to combat these uncertainties. An inventory profile illustrating safety inventory as a buffer against supply uncertainty can be seen in Figure 4, where the notation from Figure 3 is used, and the following notation is added.

- R = Reorder point: Level of inventory at which a replenishment order is placed
- S = Safety level: Minimum level of inventory that will exist in the supply chain if demand is equal to predicted demand and supply is certain
- t^* = Expected order arrival time for Cycle 2

Safety inventory is needed to satisfy demand when the supply chain faces uncertainties, such as the one shown in Figure 4. Without this safety stock on hand, demand cannot be satisfied, leading to major problems between a company and its customers.

Furthermore, work in progress (WIP) inventory, or pipeline inventory, consists of a company's products that are still traveling through the supply chain in a non-productive form [16]. This type of inventory encompasses a wide range of products and can be thought of as the inventory in the supply chain existing between the raw materials and the finished goods.

Another type of inventory is pre-build. In most supply chains, it is impossible to produce an unlimited quantity of goods at a time. However, the level of demand is not subject to these same restrictions. Thus, in order to combat these restrictions on production capacity, pre-build inventory is built up over time to satisfy future demand [17]. Seasonal inventory, which is built up during the off-season to satisfy heightened demand during the peak season, is pre-built inventory [16].

Finally, waste, also called obsolete stock, is the inventory that has exceeded its

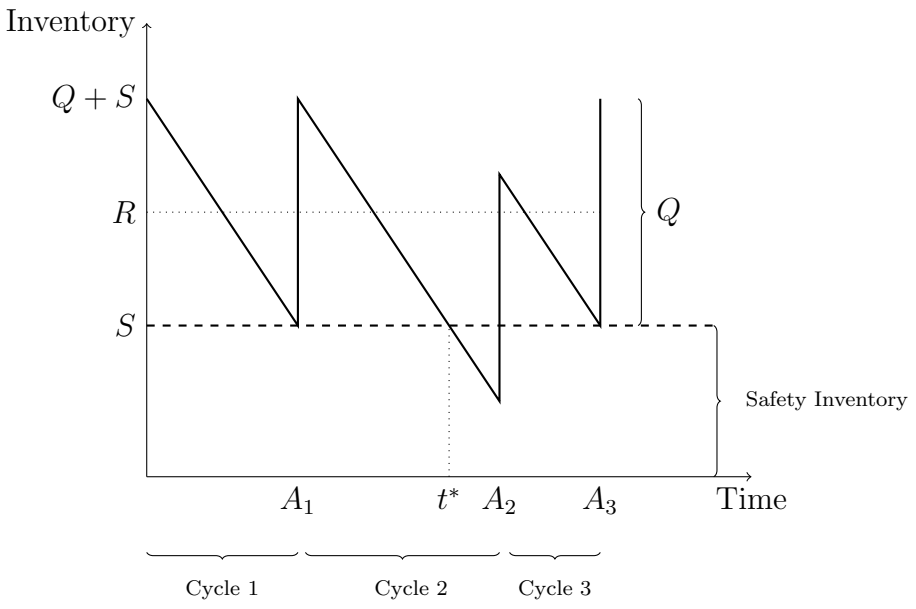


Figure 4: Inventory profile illustrating safety inventory as a buffer against supply uncertainty: In Cycle 1 and Cycle 3, inventory does not fall below the safety level, S, so cycle inventory is sufficient to satisfy demand. In Cycle 2, the replenishment order should have arrived at time t^*, but it does not arrive until time A_2. This delay in the order arrival, a type of supply uncertainty, causes the inventory to fall below the safety level, so safety inventory must be used to fill demand from time t^* to time A_2. If no safety stock had been held in this supply chain, a stock out would have occurred at time t^*.

life cycle, been damaged, or failed to sell and can no longer be kept available for sale. Obsolete inventory builds up due to uncertainty in supply and demand and lack of flexibility in the supply chain [9]. This type of inventory is useless to companies and counts as a loss on the balance sheet. In inventory optimization, waste should be

kept to a minimum.

1.3 Inventory Optimization

A very practical and potentially profitable goal is finding an optimal way to manage inventory, specifically safety stock, within a supply chain. Inventory management revolves around finding the right balance between what companies want, what companies do not want, and what companies need. Companies want to save money and maximize profit. Inventory allows companies to save money by taking advantage of economies of scale [2]. Holding inventory also helps to prevent shortage costs, which are the costs of being unable to satisfy demand, including lost profits and lost customers [7].

On the other hand, looking at inventory as an "idle resource" seems to imply that inventory should not be held because it is simply tying up money [16]. According to [7], the presence of inventory introduces the following costs.

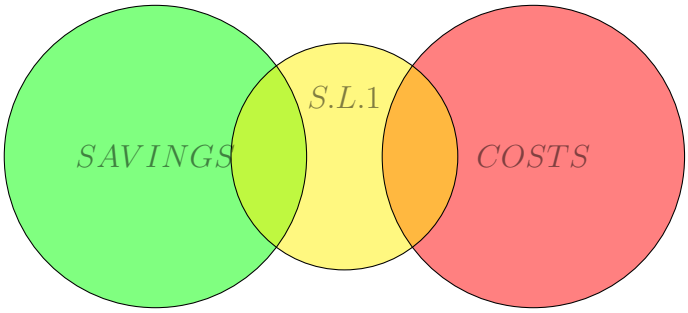
- Setup costs are the costs necessary to get everything ready to begin production, including paperwork, obtaining materials, and preparation of equipment.
- Ordering costs are the costs necessary to prepare the order itself, including calculating order quantities and managing the tracking of orders.
- Holding costs are the costs of keeping inventory on hand, including handling fees, insurance, and costs related to storage facilities.

Tying up capital on product that is sitting idle in a warehouse is not necessarily appealing from a business perspective. These costs add up quickly and are the main

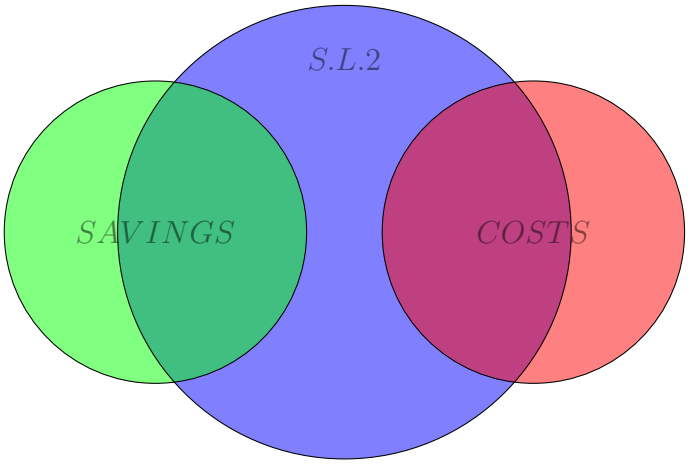
source of hesitancy when it comes to holding inventory. However, without inventory, companies have no way to combat the uncertainties they face, which makes it difficult to provide the level of service that is expected by customers. These uncertainties are numerous and include demand variability, lead-time variability, material shortages, and time-lags in order replenishment [16]. All of these wants, costs, and needs must be considered. Holding too much inventory in an attempt to increase savings through large lot sizes leads to higher holding costs and oftentimes inventory that becomes obsolete. However, trying to avoid holding costs and waste and holding too little inventory leads to stock outs, service levels that are unfulfilled, and unhappy customers.

It is significant to note that these wants, costs, and needs are not mutually exclusive, or disjoint. First and foremost, companies must focus on providing some pre-determined level of service in order to maintain business. To do this, companies hold inventory. By holding inventory, companies incur setup, ordering, and holding costs. However, at the same time, holding inventory decreases shortage costs and increases savings through economies of scale. The complexity of these relationships is what makes inventory optimization such a challenge. An illustration of this connectedness can be seen in Figure 5.

The goal of inventory optimization is to find the optimal balance among the service level, savings, and costs. Levels of cycle and safety inventory are the major factors in finding this balance. Companies can typically use past data and forecasting to get a fairly good idea of how much cycle inventory they need since it is proportional to lot



(a) Service Level 1



(b) Service Level 2

Figure 5: Relationship between service level, savings, and costs: Providing either Service Level 1 or 2 incurs costs associated with inventory holding but also savings, denoted by the overlaps in both Subfigure 5(a) and Subfigure 5(b). However, providing a higher level of service, as illustrated in Subfigure 5(b), results in higher costs but also higher savings, as shown by the greater overlap areas.

size [2]. However, figuring out how much safety inventory should be kept is much more difficult since it is affected by both supply and demand uncertainty as well as the desired customer service level [17]. This relationship can be seen in equation

(1), which is given below in Section 2.1. Supply uncertainty includes uncertainty in supply quantity or quality, uncertainty in supply timing (production lead time), uncertainty in purchase price, transportation delays, and more [2, 13]. Demand uncertainty stems from several factors, including seasonality, unsophisticated forecasting methods, forecasting errors, and lack of communication within the supply chain [2].

1.4 Simulating the Supply Chain Using SimPy

According to [14], simulation is defined as "the art and science of constructing models of systems for the purpose of experimentation," and a system is defined as "a collection of mutually dependent components whose actions on each other form a dynamic process". This perfectly describes a supply chain because downstream stages are dependent on upstream stages, and when they are combined, they form the dynamic process of the procurement and fulfillment of customer orders for some product. Because of this dependence among stages, it is often hard to predict when events within the supply chain will occur, so simulation is a tool that can be used to get a better idea of exactly what is taking place within the specific supply chain being analyzed. Our goal is to determine the optimal placement of safety stock using this simulation model.

There are many options when it comes to running simulations. One such option is SimPy, which is a Python package for "process-oriented discrete-event simulation" [8]. In the case of inventory optimization, discrete-event simulation (DES) refers to the fact that the variables of interest, namely safety stock levels, are discrete variables. Process-oriented means that every activity in the simulation is modeled as

a process, and there are multiple "application-specific threads" and a "general thread to manage the event set" [8]. Process-oriented simulations run more quickly than activity-oriented simulations, which step through time in small increments and check for the occurrence of events at each increment of time [8] For example, a year-long simulation of a supply chain would not necessarily run quickly using an activity-oriented approach. Event-oriented simulations, on the other hand, focus on the events themselves and take "shortcuts" from the scheduled time of one event to the next, which allows the simulation to run in less time [8]. SimPy's "threads" are Python generator functions. This means yield statements can be used to exit and re-enter a function at a designated point in time [8]. SimPy also has built-in shared resources, which have capacities, levels, and useful built-in statements like "put" and "get". Another advantage of SimPy is the built-in environment. The environment works as a scheduler for the events that are passed to it, and an internal clock is used to keep track of the passing of time.

Our goal is to write a SimPy simulation model that can be used to provide insight and improve upon the existing multi-echelon inventory optimization methods being used. Customer orders and the order processors within the supply chain will be modeled using generator functions. The variables of interest, safety stocks, will be modeled using shared resources, and "put/get" statements will be used to adjust inventory levels. The statistics of interest for these resources will also be monitored. Timeouts will be yielded to simulate lead-times. Finally, the simulation and monitors will be used to analyze the existing methods, and the results will be discussed.

2 MATHEMATICAL FORMULAS, DISTRIBUTIONS, AND MODELS

2.1 Safety Stock Formula

An equation for the level of safety stock that should be held, given a desired cycle service level, is given in [2]. It is assumed that both the replenishment lead time from a supplier and the customer demand are normally distributed, and a continuous review replenishment policy is used. We will use the following notation in equation (1) below, which gives the level of safety stock, SS, a company should hold.

- CSL is the desired service level the company wants to provide to customers.
- Z is the Z-score based on the standard normal distribution. A standard normal distribution is used because we are assuming demand and lead time are normally distributed. Here, $Z = F^{-1}(CSL)$, where F^{-1} is the inverse of the standard normal.
- D is the average demand per period.
- σ_D is the standard deviation of demand per period.
- L is the average replenishment lead time.
- σ_L is the standard deviation of replenishment lead time.

$$SS = Z \cdot \sqrt{L{\sigma_D}^2 + D^2{\sigma_L}^2} \tag{1}$$

Equation (1) illustrates safety stock's dependence on both the uncertainty in customer demand, through σ_D, as well as the uncertainty in the production lead time of

the product, through σ_L. This dependence on factors that are uncertain makes determining the optimal level of safety stock a very difficult task. Our goal is to analyze the usefulness of this formula in multi-echelon inventory optimization and determine why the formula provides more accurate safety stock requirement calculations in some situations than others based on the SimPy simulations.

2.2 Related Probability Distributions

There are several probability distributions that will be used in the implementation of the SimPy model. Equation (1) in Section 2.1 incorporates the standard normal distribution. The normal (or Gaussian) distribution is a continuous probability distribution known for its "bell" shape that is centered at the distribution mean μ with standard deviation σ, and the standard normal distribution is a special case in which $\mu = 0$ and $\sigma = 1$ [11]. Referencing [5], the probability density function for a random variable X over the real line that is normally distributed with mean μ and standard deviation σ is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}. \tag{2}$$

As discussed previously, implementation of the safety stock formula given in equation (1) assumes that both the replenishment lead time and customer demand are normally distributed random variables. This assumption of normality is significant and must be kept in mind when using equation (1).

The Poisson distribution is used in the model in more of an indirect way. The Poisson distribution is a discrete distribution used to model the number of occurrences of some event during an interval of time [11]. The event occurrences are assumed to be

independent of one another, and the mean of the distribution is traditionally denoted λ, which represents the average number of event occurrences in one time period [11]. Referencing [5], if X is a random variable over the non-negative integers, then it is Poisson distributed with mean $\lambda > 0$ if its probability distribution is

$$p(x) = \frac{\lambda^x}{x!} e^{-\lambda}. \tag{3}$$

Customer arrivals are discrete event occurrences that are independent of one another, so they can be modeled as a Poisson process with mean λ representing the average number of customer arrivals per day.

An alternate option for modeling customer arrivals with the Poisson distribution is to use the Exponential distribution. The Exponential distribution is a continuous distribution that can be used to model the time intervals between event occurrences in a Poisson process [11]. This means that the Exponential distribution can be used to model the time between the customer arrivals instead of modeling the arrivals themselves. Referencing [11], the probability density function for a random variable X over the non-negative reals that is Exponentially distributed with scale parameter $\theta > 0$ is

$$f(x) = \frac{1}{\theta} e^{-x/\theta}. \tag{4}$$

According to [1], for a Poisson process with mean rate λ, the length of the time period from some fixed time to the next occurrence of an event has the probability density function

$$f(x) = \lambda e^{-\lambda x}. \tag{5}$$

Setting $\theta = 1/\lambda$ in equation (4) yields equation (5). This means that the length

of the interval from some fixed time, say the time of an occurrence of the event, to the next occurrence of the event (i.e. the interarrival time) has an Exponential distribution with mean $\theta = 1/\lambda$. Thus, since customer arrivals can be modeled as a Poisson process with an average rate of λ arrivals per day, the interarrival times can be modeled using an Exponential distribution with mean $1/\lambda$.

2.3 Graves and Willems Model

Some of the most widely used and referenced models in inventory optimization are the single-stage and multi-stage models presented by Graves and Willems in [4]. These models assume the following:

- The supply chain can be modeled as a network.

  Stages in the supply chain are represented as nodes. A stage is anywhere in the supply chain that some type of work on the product occurs. Arcs represent the existence of a relationship between an upstream stage and a downstream stage. In a supply chain with r total stages, suppose that u of those stages are upstream stages. This implies there are $r - u$ downstream stages. Upstream stages are denoted with the letter i, where $i = 1, ..., u$, and downstream stages are denoted with the letter j, where $j = u + 1, ..., r$. If an upstream stage is a supplier to a downstream stage, an arc exists between the two stages. Each arc has a corresponding scalar ϕ_{ij}, which indicates downstream stage j needs ϕ_{ij} units of material from upstream stage i to produce one unit of product.

- The production lead-time, T_j, for each stage is known and deterministic.

The production lead-time begins when all necessary inputs are available to begin production at a stage and ends when the final product is ready to be used to satisfy demand. This model further assumes that lot size has no effect on the production lead-time.

- A periodic-review base-stock replenishment policy is used at every stage.

  This policy involves reviewing inventory periodically, as opposed to continuously, and ordering enough product at regular intervals to increase on-hand inventory to some specified base-stock. This model also assumes that all stages share a common review period.

- Both end-item demand and internal demand are bounded.

  In this model, end-item/external demand is only seen at stages that have no successors. These stages are called demand nodes. It is assumed that the end-item demand at stage j, $d_j(t)$, comes from a process where we can determine the average demand per period μ_j. These demand nodes place orders to upstream, or internal, stages. The demand at an internal stage, $d_i(t)$, is simply the sum of orders received from the demand nodes. Thus, we have

$$d_i(t) = \sum_{(i,j)\in A} \phi_{ij} d_j(t) \tag{6}$$

  where A is the set of all arcs in the network. This model assumes that the demand at every node in the network is bounded. End-item demand is bounded by the function $D_j(\tau)$, for $\tau = 1, 2, ..., M_j$, where M_j represents the maximum replenishment time for stage j. Thus, for any period of time t and a replen-

ishment time of τ, the demand at stage j is bounded above by $D_j(\tau)$, as given by

$$d_j(t-\tau+1) + d_j(t-\tau+2) + ... + d_j(t) \leq D_j(\tau). \quad (7)$$

Note that no assumptions are made about the distribution of demand for this model. Inderfurth [6], on the other hand, assumes normally distributed demands.

- Every stage ensures a guaranteed service time in which it will fill all customer demand.

  Under this model, demand nodes quote a guaranteed service time of S_j to customers, and internal stages quote a guaranteed service time of S_{ij} to downstream stages. It is assumed that an internal stage quotes the same guaranteed service time to all downstream stages, so we set $S_{ij} = S_i$ for each demand node j. It is assumed that these guaranteed service times are not violated; each stage provides 100% service. This assumption is the focus of Section 1.3.

2.3.1 Single-Stage Model

Single-stage models are very important in that they are the basis for multi-stage models. A single-stage supply chain can simply be thought of as a demand node. There are no upstream stages leading into the demand node, so upstream service times do not come into play in single-stage models. This allows us to model a single stage, which we can then use as a building block to model multi-stage supply chains.

Graves and Willems [4] define the single-stage model as follows. The inventory

profile begins at time $t = 0$ with initial inventory $I_j(0) = B_j$, which is the base stock at stage j. The level of finished inventory at the end of period t at stage j depends on:

- the base stock, denoted B_j,
- the amount of inventory that must be used to fill orders during that period, denoted $d_j(0, t - S_j)$, where S_j is the guaranteed service time at stage j, and
- the amount of product that has been replenished to be added to existing inventory, denoted $d_j(0, t - SI_j - T_j)$, where SI_j is the inbound service time at stage j and T_j is the production lead time at stage j.

The difference between the replenishment to inventory and the use of inventory to fill orders is called the inventory shortfall, denoted $d_j(t - SI_j - T_j, t - S_j)$. The inventory at stage j at time t is the base stock minus the inventory shortfall as given by

$$I_j(t) = B_j - d_j(t - SI_j - T_j, t - S_j). \tag{8}$$

An explanation of equation (8) begins with the fact that the inventory at time t is built upon the base stock, B_j. At time t, orders through time $t - S_j$ have been filled since a service time of S_j is guaranteed. Under this assumption, orders placed after time $t - S_j$ are not required to be filled by time t. Also at time t, demand through time $t - SI_j - T_j$ has been replenished since it takes $SI_j + T_j$ to fully complete production. Note that demand after time $t - SI_j - T_j$ will not be replenished until after time t. Graves [4] calls this period from time $t - SI_j - T_j$ to time $t - S_j$ where demand must be covered from inventory, specifically base stock, a "time interval of exposure". The

amount of inventory used to satisfy demand during this time interval is the quantity that is subtracted from the base stock, B_j, to obtain equation (8). The length of the "time interval of exposure" is what is known as the net replenishment time.

In order to satisfy the guaranteed service time assumption, we must have $I_j(t) \geq 0$ for each stage j. Following from equation (8), this requirement implies

$$B_j \geq d_j(t - SI_j - T_j, t - S_j). \tag{9}$$

Recall that the goal of inventory optimization is to satisfy demand while minimizing the costs associated with inventory holding. Demand at stage j is bounded by $D_j(\tau)$ as seen in equation (7). Base stock must be able to cover the demand during the net replenishment time, so we set the base stock as the maximum possible demand during the net replenishment time using

$$B_j = D_j(\tau), \tag{10}$$

where $\tau = SI_j + T_j - S_j$ is the net replenishment time for stage j. This allows us to satisfy equation (9) with the least amount of inventory possible.

Furthermore, the safety stock at stage j can be computed as the expected inventory level, $E[I_j]$, at that stage where

$$\begin{aligned} E[I_j] &= E[B_j - d_j(t - SI_j - T_j, t - S_j)] \\ &= D_j(\tau) - ((t - S_j) - (t - SI_j - T_j))\mu_j \\ &= D_j(SI_j + T_j - S_j) - (SI_j + T_j - S_j)\mu_j. \end{aligned} \tag{11}$$

Finally, it is assumed that the demand rate and production lead times are inputs, so the WIP inventory is predetermined since it only depends on the production lead time.

2.3.2 Multi-Stage Model

According to Graves and Willems [4], the multi-stage inventory model uses the single-stage model at each stage and requires the inbound service time at a stage to be a function of the upstream service times. Requirements on the net replenishment time and the inbound service time are added to the single-stage safety stock equation to give the following model for stage j:

$$E[I_j] = D_j(SI_j + T_j - S_j) - (SI_j + T_j - S_j)\mu_j, \tag{12}$$

$$SI_j + T_j - S_j \geq 0, \tag{13}$$

$$SI_j - S_i \geq 0 \text{ for all } (i, j) \in A. \tag{14}$$

As in the single-stage model, equation (12) represents the safety stock at stage j. Equation (13) guarantees a non-negative net replenishment time. Equation (14) ensures that the inbound service time at stage j is at least as great as each of the upstream service times. This model further assumes that the maximum service times for the demand nodes, the production lead times, and the means and bounds of demand are all known input parameters. This model makes a very strong assumption in that a guaranteed service time is quoted to each customer. As a customer, this is a welcomed promise. However, as a company using this model, this is a strong guarantee to have to fill. When going from a single-stage to multi-stage supply chain, the interaction between the upstream and downstream stages becomes the focus. Each downstream stage in the supply chain relies almost completely on upstream stages. For example, a breakdown upstream delays production, which delays delivery to its successive stage(s), which delays production, and so on. In a simple two-stage

supply chain, the lead time error at stage 2 is a function of the service level at stage 1. This becomes very complicated when more and more stages are involved because the effects of one small problem upstream will trickle all the way down the supply chain to the demand nodes where a certain service time is guaranteed to every single customer. Because of this interdependence among stages, safety stock is a vital factor in being able to provide this guaranteed service time.

2.4 Guaranteed Service Time

Under the assumption of guaranteed service time, all demand nodes promise to satisfy 100% of customer demand in some set time, S_j, and all internal stages promise to satisfy 100% of downstream demand in some set time, S_i. Guarantees of total reliability carry huge implications and responsibilities, and this one is no exception. Although both safety stock and operating flexibility are used to guarantee this 100% service level, it is a difficult task to coordinate the use of multiple methods at the same time [6].

On the surface, a guaranteed service time is simply a set amount of time in which customer demand must be satisfied. In other words, if demand node j guarantees a service time of S_j, external demand at time t must be satisfied by time $t + S_j$ [4]. More specifically, service time is a function of general processing time, load, and supply unreliability. General processing time is essentially the time it takes to process the order and fill out the necessary paperwork. This amount of time is fixed and usually pretty short. Once this process is complete, manufacturing begins. The load can be thought of as the manufacturing lead time, which equates to the service

time upstream. Finally, supply unreliability must be considered. Unforeseen problems upstream, such as a machine breakdown, lead to an unexpected decrease in supply. When setting inventory levels, we must account for the machine breakdown as well as the production that is lost during the breakdown. To account for this unexpected decrease in supply, companies need to have a buffer (safety stock) on hand. All of these components must be taken into account when trying to guarantee a service time, with the main focus being the unreliability in supply. Our goal is to use the SimPy simulations to get an idea of how the safety stock equation performs because having the proper amount of safety stock on hand is a key factor in being able to assume a guaranteed service time.

3 THE SIMPY SIMULATION MODEL

For the purposes of this model, we consider a multi-echelon supply chain consisting of a supplier, a bulk container, a drum, and external customers. The code listing given in Appendix A is specific to this supply chain; however, a SimPy simulation model such as this one can be implemented for other supply chains as well. This section describes how a model such as this one can be designed and implemented for a general multi-echelon inventory optimization problem.

3.1 Introduction to Simulation Software and Model

To implement the simulation model given in Appendix A, the Anaconda Python distribution is used. The first step in creating the simulation model is to import all necessary Python packages and fuctions. For the simulation and monitoring, SimPy and NumPy are imported. SimPy, as discussed in Section 1.4, is a simulation tool that is accessed as a Python package. SimPy 3 is used to run the model in Appendix A. SimPy 3 is an update to SimPy 2 that is written for Python 3 and is a general purpose modeling tool; whereas SimPy 2 is limited in the types of simulations that can be implemented. Fewer imports are needed in SimPy 2 than SimPy 3 [15]; however, some NumPy functions are indeed used and must be imported. As discussed in Section 2.2, the Exponential distribution is used to generate customer inter-arrival times because the arrivals themselves are assumed to follow a Poisson process. The random number generator is seeded with a positive integer to allow the simulation to be reproduced. The Math package is used for monitoring and computing statistics of interest from the simulation. The Repeat function is used to create lists where these monitored

statistics can be stored.

The next step in designing a model such as the one in Appendix A is to define the global parameters for the supply chain. By defining the parameters in a class, their values can be changed easily for analysis purposes. Next, the elements of the simulation used for monitoring are defined in another class. These elements will be reset for each simulation run. Depending on what is being modeled, the appropriate elements should be created in this class.

3.2 Components of the Model

There are four main components in this specific supply chain: inventory, orders/-customers, the bulk container node, and the drum node. Each of the components is modeled using a SimPy class, process, or container. Python documentation defines classes as objects that "normally act as factories for new instances of themselves" [12]. Certain properties are defined for each class, and when a new instance of the class is created, these properties must be given. For example, the customer class in this example includes the customer name, action, and environment as its properties. Processes are used to model "the behavior of active components" of the simulation [15]. For example, customer order placements are modeled as processes. Containers are a type of SimPy resource used to model the sharing of some homogeneous material between processes [15]. In this case, the homogeneous material is inventory, which is measured in units. Containers have built-in put/get statements used to increase/decrease their level, which level can always be checked.

3.2.1 External Components

The model's external components are the supplier (denoted SU) and the customers (denoted C), which can be split into bulk container (BU) customers and drum container (DR) customers. The supplier is the source of supply for the bulk container, which means replenishment orders are placed from the bulk container to SU. External customers can place orders to BU as well as to DR. For this simulation, we are assuming that customers arrive to both DR and BU according to (separate) Poisson processes.

3.2.2 Internal Components

The model's internal components are the bulk container (denoted BU) and the drum (denoted DR). BU is supplied directly from the supplier, and DR is supplied from BU. When an external customer places an order to DR and the inventory on-hand is not sufficient ot fill the order, DR places an order to BU for the size of the order, which is filled when the inventory level at BU becomes sufficient to do so.

By combining the internal and external components and the relationships between them, we get the supply chain illustrated below in Figure 6.

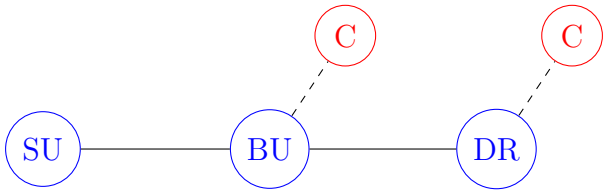


Figure 6: Illustration of the supply chain being used to construct the simulation model given in Appendix A. The stages of the supply chain are shown in blue, and external customers are shown in red. Solid lines are used to illustrate the flow of products through the supply chain itself, and dashed lines illustrate the the flow of products from the supply chain to external customers.

3.3 Implementation of BU Node

As Figure 6 illustrates, BU is connected to external customers, the supplier, and the drum container. When the BU inventory level reaches the designated re-order point, BU places a replenishment order to SU. BU must satisfy demand from both external customers and DR. When an order at DR cannot be filled, BU receives an order from DR that must be filled as soon as possible. Table 1 describes the entities related to BU, grouped by class.

Table 1: Description of model entities related to BU node by class, including name, type, function, and all related parameters

Class Name			
Entity	Type	Function	Related Parameters
Inventory			
BU_inv	Container	Models inventory on-hand at BU node	• P.ROP_BU: initial level of BU_inv
mon_procBU	Process	Monitors inventory level at BU node daily and triggers replenishment orders to supplier when designated inventory level is reached	• P.ROP_BU: level at which replenishment order is placed • P.Q_1: lot size of replenishment orders • P.LT_1: replenishment lead time between BU and supplier
BUCustomer			
ordertoBU	Process	Models the action of a BUCustomer, which is the placement of an order to BU	• P.BUorderLotSize: size of order placed by BUCustomer
BUOrderProcessor			
BUEntrance	Process	Models the arrival of external customers to BU using Exponentially distributed interarrival times	• P.externalToBUMean: daily mean of Poisson process that models arrival of BU customers • P.simulationTimeMax: time at which customer arrivals cease

3.4 Implementation of DR Node

Unlike BU, DR is only connected to external customers and the bulk container. DR is supplied from BU, and DR is solely responsible for supplying product to external DR customers. Table 2 describes the entities related to DR, grouped by class. The only significant differences between DR and BU, other than different parameter values, are that DR places replenishment orders to BU instead of SU, and when DR cannot fill a customer order, an order is automatically placed to BU for the size of

the customer order.

Table 2: Description of DR model entities by class, including name, type, function, and all related parameters

Class Name			
Entity	Type	Function	Related Parameters
Inventory			
DR_inv	Container	Models inventory on-hand at DR node	• P.ROP_DR: initial level of DR_inv
mon_procDR	Process	Monitors inventory level at DR node daily and triggers replenishment orders to BU when designated inventory level is reached	• P.ROP_DR: level at which replenishment order is placed • P.Q_2: lot size of replenishment orders • P.LT_2: replenishment lead time between DR and BU
DRCustomer			
ordertoDR	Process	Models the action of a DRCustomer, which is the placement of an order to DR	• P.DRorderLotSize: size of order placed by DRCustomer
DRorderToBU	Process	If DR inventory is not sufficient to fill a customer order, this process models the order placed by DR to BU to fill the customer order	• P.DRorderLotSize: size of order placed by DR to BU • P.LT_2: lead time between DR and BU
DROrderProcessor			
DREntrance	Process	Models the arrival of external customers to DR using Exponentially distributed interarrival times	• P.externalToDRMean: daily mean of Poisson process that models arrival of DR customers • P.simulationTimeMax: time at which customer arrivals cease

3.5 Monitoring Within the Simulation

The ultimate goal of this simulation is to gather information that can be used to analyze existing multi-echelon inventory optimization (MEIO) methods. In order to

gather such information, the components of the simulation must be modeled. Monitors will differ from problem to problem, depending on the nature of the problem and the goal of the simulation. As with the discussion of model components, the discussion of monitoring can be broken down into internal and external monitoring. For the supply chain being modeled here, an inventory object and empty lists for both BU and DR customer wait times are created. These lists will be used to store wait times, which we ultimately want to minimize. Counts/tallies, such as the number of customer or orders, can also be used for monitoring purposes.

3.5.1 External Monitoring

External monitoring involves collecting data and computing several different statistics relating to the customers/orders. In relation to Figure 6, these monitors are gathering information from the two nodes labeled C. First, lists of wait times per order are recorded for BU and DR customers. When a customer places an order, the current time is recorded and is used to manually monitor the customer wait times once the order is fulfilled. Second, the number of customer/order arrivals is recorded for both BU and DR. This is done through a count that starts at zero and is increased by one each time a customer arrival is simulated.

3.5.2 Internal Monitoring

Internal monitoring involves keeping track of inventory levels and demand at BU and DR. In relation to Figure 6, these monitors are gathering statistics at the BU and DR nodes. The inventory level monitors are executed through the processes described

in Tables 1 and 2. These monitoring processes keep track of the inventory levels and trigger the placement of replenishment orders when the designated re-order point is reached. The demand monitors gather daily demand counts, which are stored in separate lists, by node, and can be used for analysis once the simulation is complete.

3.6 Running the Simulation

In order to run the simulation, we must initialize the components of the model that have been previously defined. To do this, we define a model process that is then called to run the simulation. A seed is used so that the simulation can be reproduced, so a seed for the random number generator is first initialized. Then, the lists of DR and BU waits are initialized as empty lists, which will be appended as the simulation runs. The environment is then created using the built-in SimPy environment feature. Next, the BU and DR order processors (nodes) are created and placed into the environment, and the corresponding lambdas used for generating inter-arrival times are defined for each. The inventory container is then created and placed into the environment. The built-in run feature is then used to run the simulation until the specified time. The while loops used in the creation of the customers refer to this specified time. Finally, the monitored statistics of interest (DR and BU wait times, DR and BU demand per day, number of DR and BU customers/orders) are returned at the end of the simulation output. This model is then called, and the simulation will run and produce the specified print statements and monitored statistics that can be used for analysis purposes.

3.7 Production of Simulation Results

As discussed previously, customer demand at both BU and DR is assumed to follow (independent) Poisson distributions, which implies BU and DR customer interarrival times are assumed to follow (independent) Exponential distributions. Using these assumptions, customer (order) arrivals are simulated, and demand per day at BU and DR is monitored. From this, average daily demand for BU and DR is calculated along with the corresponding standard deviation of daily demand. Then, using the safety stock equation given in equation (1), safety stock requirements are calculated first based on the assumed demand and second based on the simulated demand. For the simulated demand, the corresponding standard deviation of demand is used in the safety stock calculation. In order to analyze the effect of the accuracy of the predicted standard deviation of demand, a range of appropriate standard deviations is used along with the assumed average demand. These safety stock calculations are made for each simulation run and for a range of cycle service levels.

4 RESULTS AND CONCLUSIONS

Up to this point, we have built a foundation and carried out the design and implementation of a SimPy simulation model to analyze safety stock levels within a simple two-stage supply chain. A common method of predicting safety stock requirements is the implementation of the safety stock equation, which is given in equation (1). However, our hypothesis is that the equation's strict distribution assumptions cause the formula to behave better in some situations than others. Specifically, our goal is to use the results provided by the SimPy simulation to analyze for what range of parameter(s) equation (1) performs well and what happens beyond this range. It is important to note that we are assuming (independent) Uniformly distributed replenishment lead times for both BU and DR, so the source of variability in our analysis comes from the standard deviations of the BU and DR daily demand. In terms of equation (1), we set $\sigma_L = 0$ so that σ_D is the source of variability we are concerned with. First, we use the results to show that the simulation model agrees with the theoretical model under the necessary conditions. Second, we analyze the safety stock equation for a range of demand standard deviations to conclude where the theoretical model breaks down. Next, we discuss some of the potential reasons for this break down and why these conclusions are meaningful. Finally, we briefly mention other possibilities and variations of the model that could be considered. Throughout this section, when a formal test is performed, a significance level of $\alpha = 0.01$ is used.

4.1 Model Validation

In order to confirm that the SimPy model is performing the way that it should be, we must show that the simulation model agrees with theoretical model when the necessary conditions are satisfied. Recall that the theoretical model carries the assumption that demand is normally distributed. For the simulation model to be valid, it should produce the same results as the theoretical model when the condition of normally distributed demand is met. Customer arrivals, which correspond to orders, are modeled using Poisson distributions. However, when the mean λ is large enough, the highly-skewed Poisson distribution becomes symmetric [11]. Thus, in order to validate the simulation model, we run the model with large λ values for the mean daily demands. Specifically, we use a mean number of orders per day for BU of 50 (order size of 20) and a mean number of orders per day for DR of 50 (order size of 10). Running five simulation runs for each node under these conditions produces plots of daily demand (in units) for BU and DR as shown below in Figures 7 and 8, respectively.

Based on Figures 7 and 8, we conclude that simulated demand with a large enough mean does appear to be normally distributed. However, a formal test can be used to verify this. The Shapiro-Wilk test is a formal test used to test if data comes from a normally distributed population [10]. For each of the five simulation runs for BU, the test returns a p-value greater than $\alpha = 0.01$. For four of the five simulation runs for DR, the test returns a p-value greater than $\alpha = 0.01$, with the fifth test returning a p-value very close to 0.01. Each of these p-values and the corresponding Shapiro-Wilk test statistic can be seen in Appendix B. These p-values, along with the plots, lead

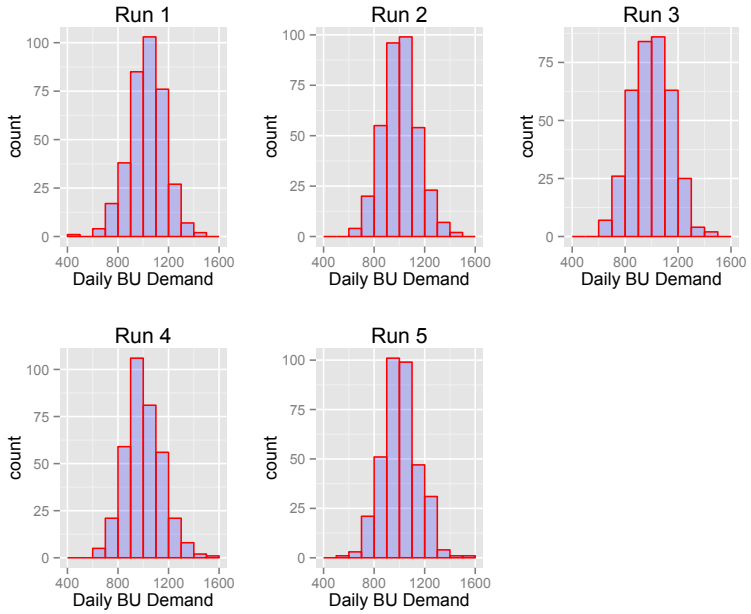


Figure 7: Plots of simulated BU demand with daily mean of 50

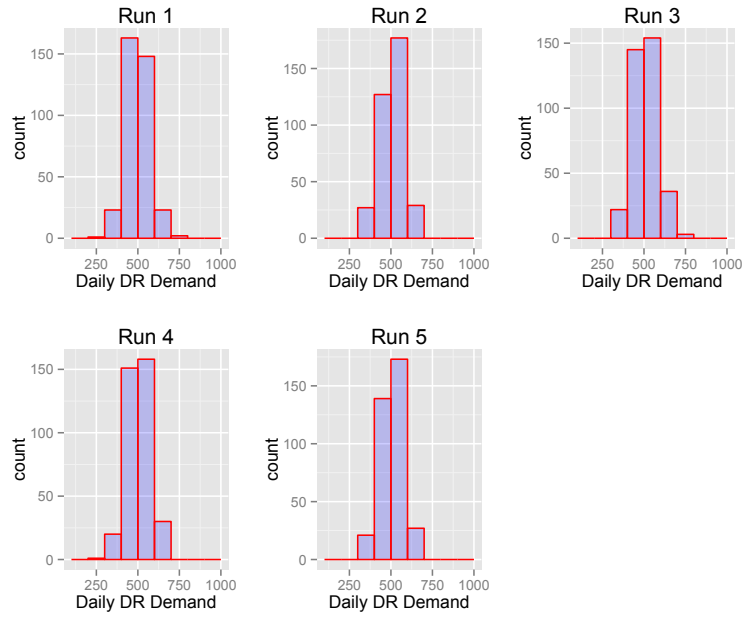


Figure 8: Plots of simulated DR demand with daily mean of 50

to the conclusion that the sample of daily demand in each simulation run comes from a normally distributed population. Thus, the simulation model produces normally distributed demand regardless of the seed chosen if the mean daily demand is large enough.

In order to confirm that the simulation model does in fact agree with the theoretical model under the necessary conditions, these five runs are used to calculate the levels of safety stock needed using equation (1). If the two models do in fact agree, the safety stock calculations for the two sets of demand should be very similar. Plots of safety stock calculations based on the assumed demand and simulated demand for each of the five runs are shown below in Figures 9 and 10 for BU and DR, respectively.

The safety stock calculations based on the assumed and simulated demands are very similar, as shown by the similarity in the red and blue lines in Figures 9 and 10. Thus, we conclude that the simulation model does in fact produce normally distributed demand when a large enough mean is used, and the simulation model does agree with the theoretical model when the necessary conditions are satisfied.

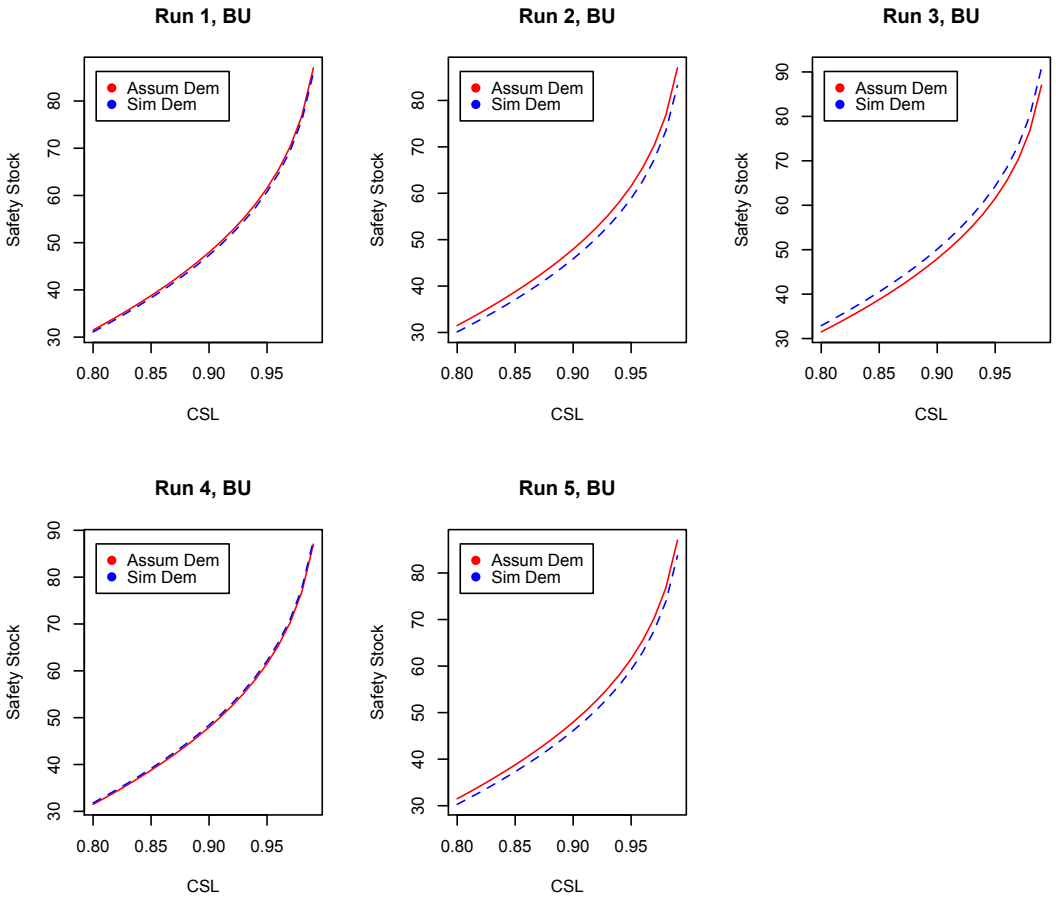


Figure 9: Plots of required safety stock by cycle service level for BU based on assumed and simulated demand

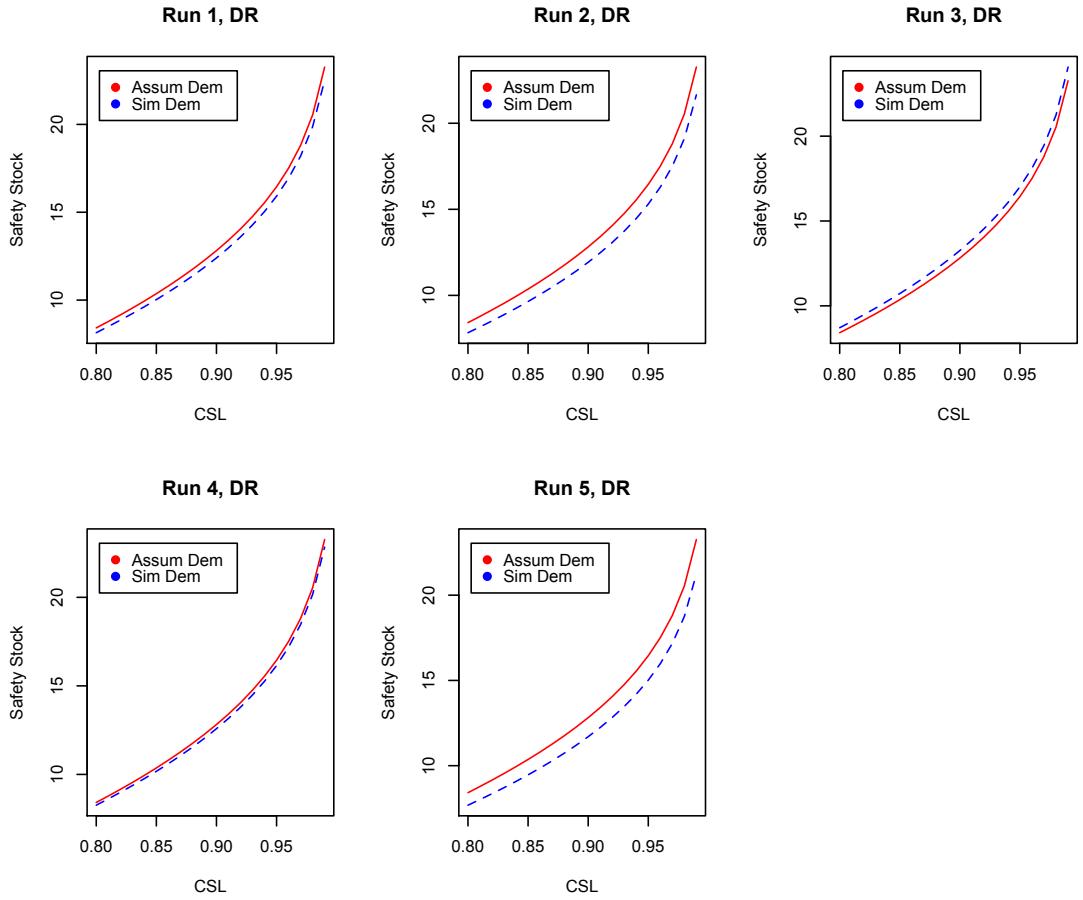


Figure 10: Plots of required safety stock by cycle service level for DR based on assumed and simulated demand

4.2 Analysis of Safety Stock Equation

Now that the simulation model has been shown to agree with the theoretical model under the necessary conditions, we want to analyze the safety stock equation when these assumptions are not met. We want to do this because the assumption of normally distributed demand is not necessarily realistic in practice. As discussed

previously, we are assuming Uniformly distributed replenishment lead times for the sake of this problem, so the source of variability comes from the standard deviation of the demand. In order to analyze the performance of the safety stock equation, we must take this into account. To do this, we will use the simulation results to calculate the safety stock requirements based on the simulated demand and corresponding standard deviation and based on the assumed demand for a given range of standard deviations. This will allow us to see how the calculations differ depending on how accurate the forecast for standard deviation of demand is. For both BU and DR, we will use five runs of simulated demand. It is important to note that the simulated demand no longer comes from a normally distributed population, which means the safety stock equation assumptions are no longer satisfied. Plots to confirm the non-normality of demand are shown below in Figures 11 and 12.

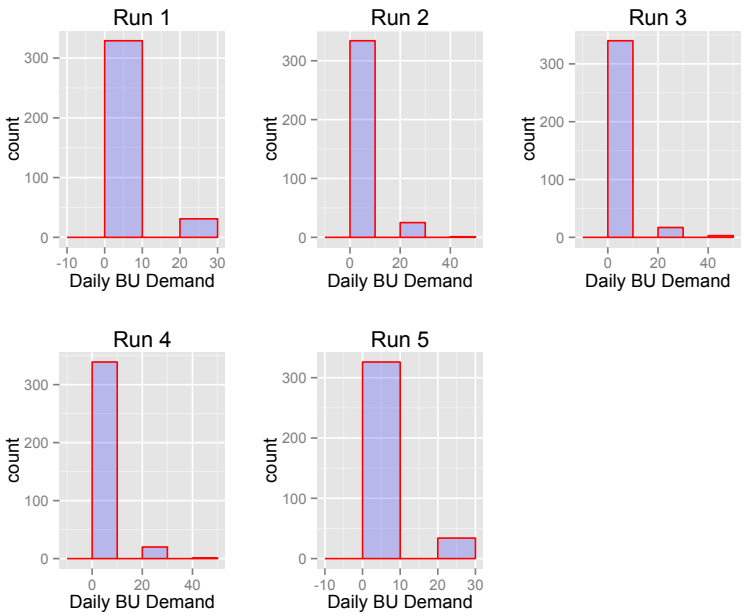


Figure 11: Plots of simulated BU demand with daily mean of $\frac{5}{60}$

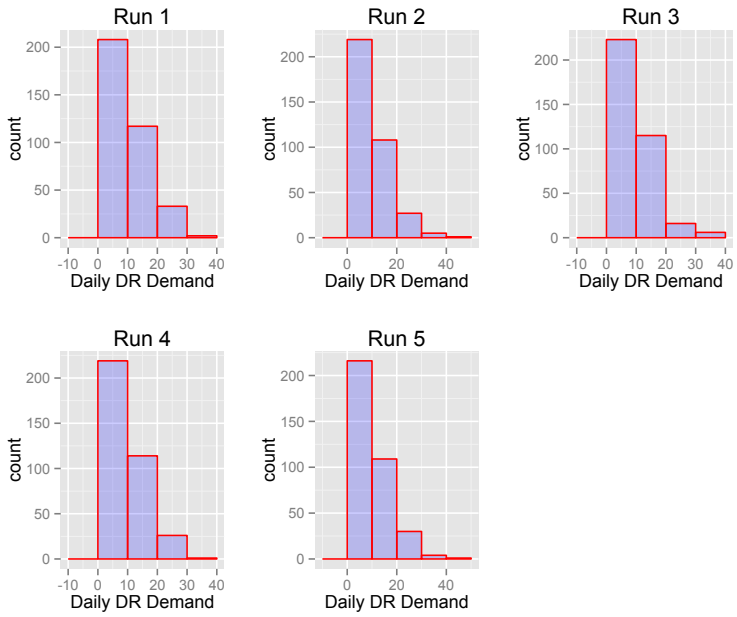


Figure 12: Plots of simulated DR demand with daily mean of $\frac{1}{2}$

The plots in Figures 11 and 12 show that the simulated demand is highly skewed as expected, but a formal test can be used to confirm this. Shapiro-Wilk test p-values less than $\alpha = 0.01$ for each of the ten simulations runs confirm this observation and are given in Appendix B. The plots of predicted safety stock requirement calculations are shown below in Figures 13-17 for BU and Figures 18-22 for DR.

Figures 13 through 22 below show how the safety stock formula performs, given the designated assumed standard deviation of demand. When the predicted standard deviation is relatively close to the simulated standard deviation of demand, equation (1) performs well for any cycle service level. However, as the predicted standard deviation of demand becomes less accurate, the safety stock formula clearly breaks down, especially as the desired cycle service level increases. It is important to note that if

the predicted standard deviations are underestimated, the safety stock requirements given by the formula are also underestimated. On the other hand, if the predicted standard deviations are overestimated, the safety stock requirements given by the formula are also overestimated.

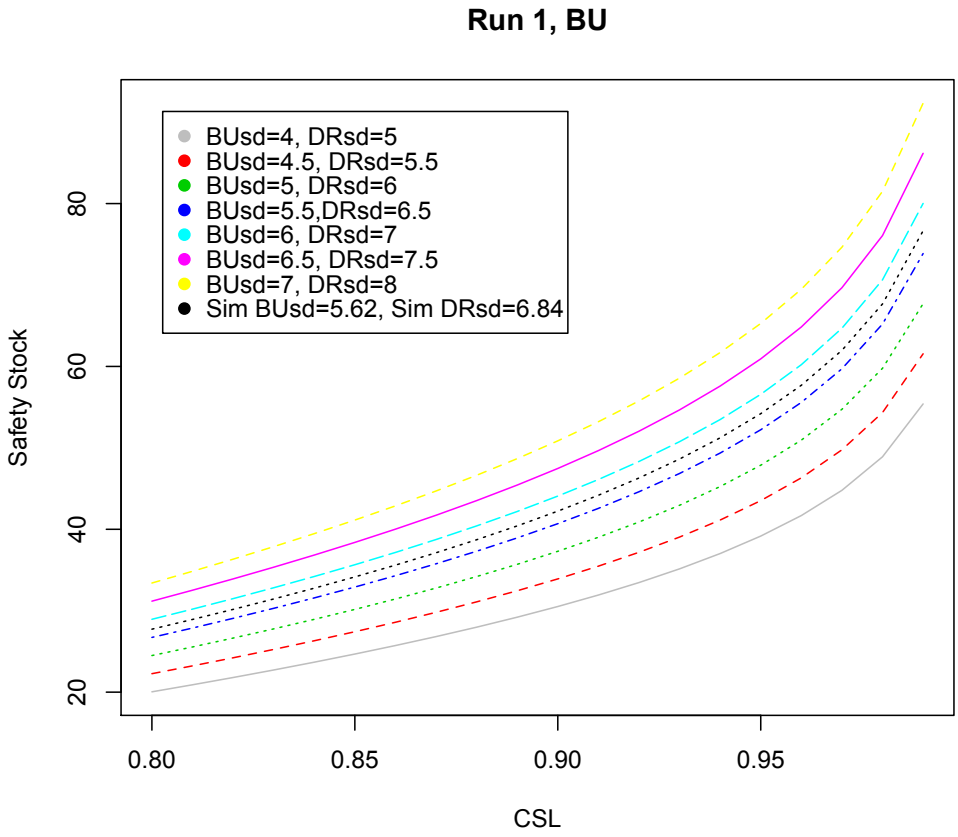


Figure 13: Plot of required safety stock by cycle service level for BU based on Run 1 simulated demand and assumed demand for a range of standard deviations

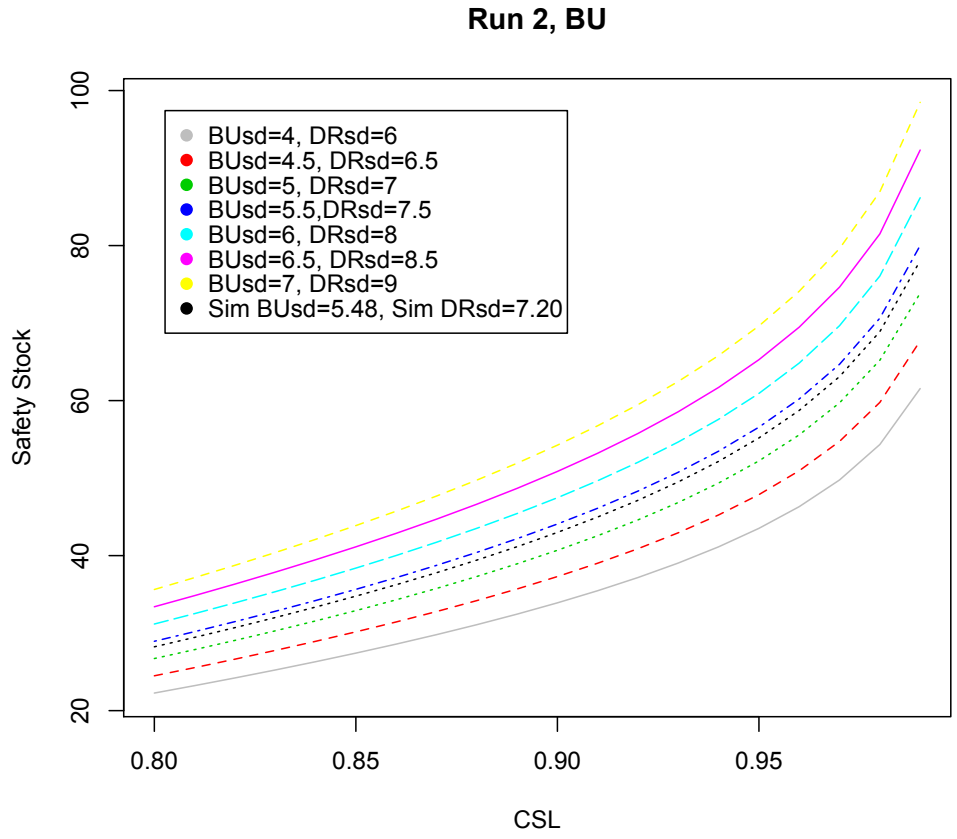


Figure 14: Plot of required safety stock by cycle service level for BU based on Run 2 simulated demand and assumed demand for a range of standard deviations

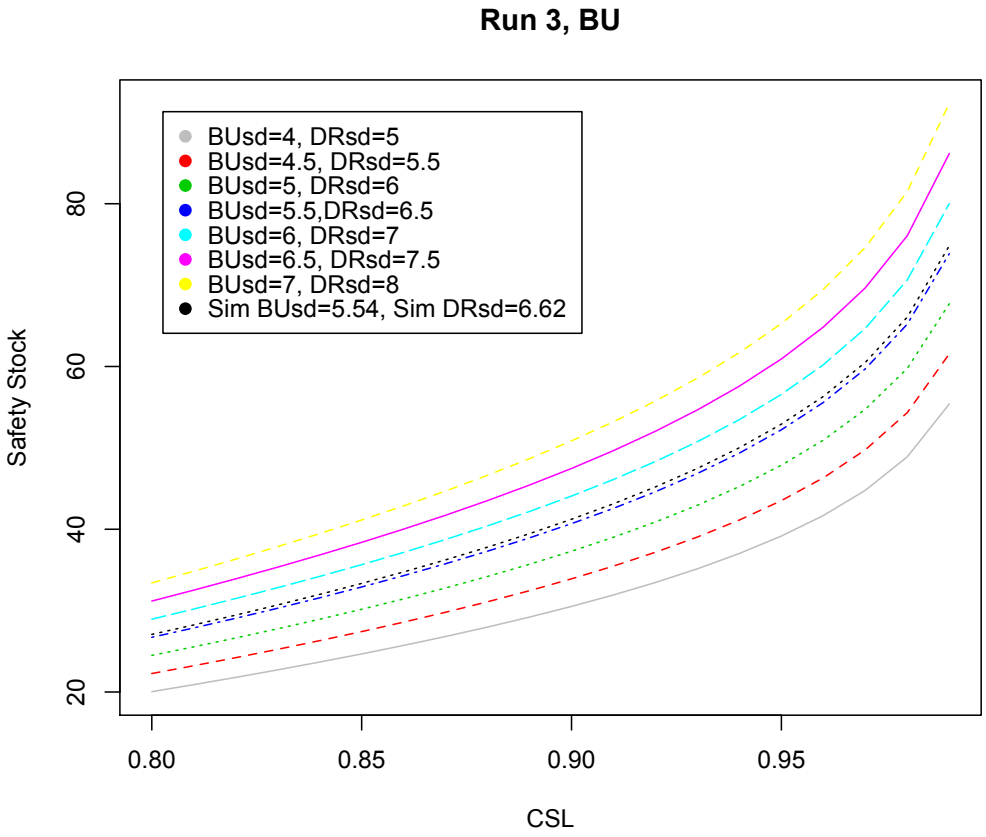


Figure 15: Plot of required safety stock by cycle service level for BU based on Run 3 simulated demand and assumed demand for a range of standard deviations

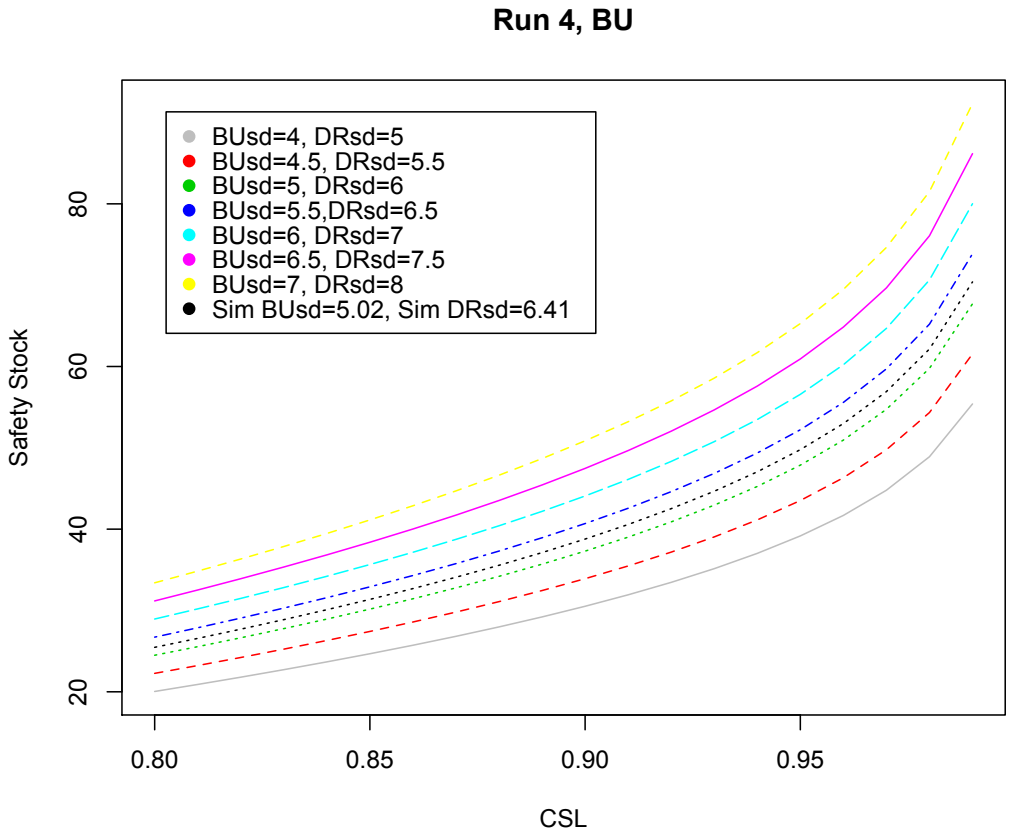


Figure 16: Plot of required safety stock by cycle service level for BU based on Run 4 simulated demand and assumed demand for a range of standard deviations

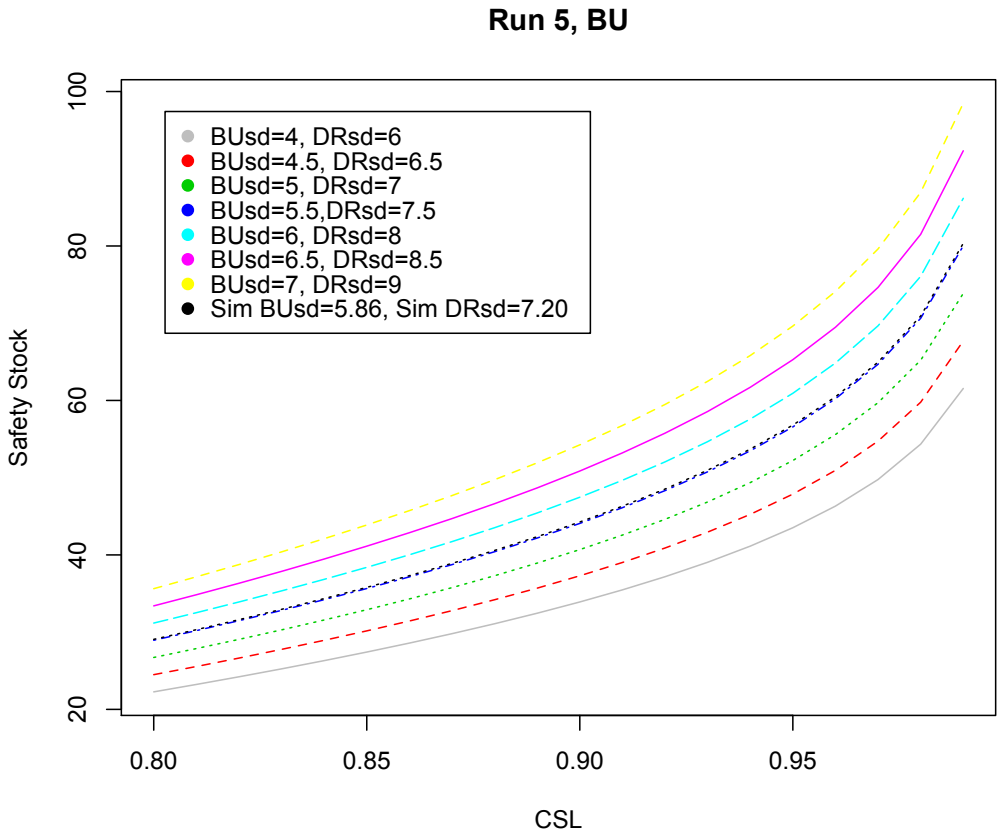


Figure 17: Plot of required safety stock by cycle service level for BU based on Run 5 simulated demand and assumed demand for a range of standard deviations

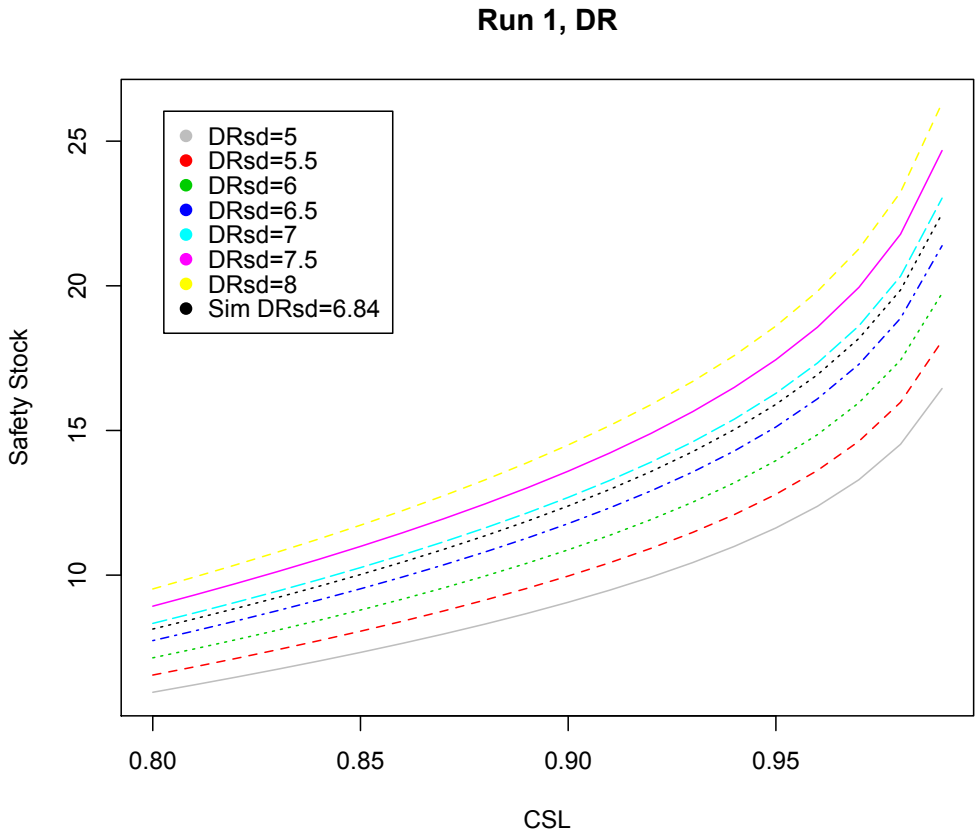


Figure 18: Plot of required safety stock by cycle service level for DR based on Run 1 simulated demand and assumed demand for a range of standard deviations

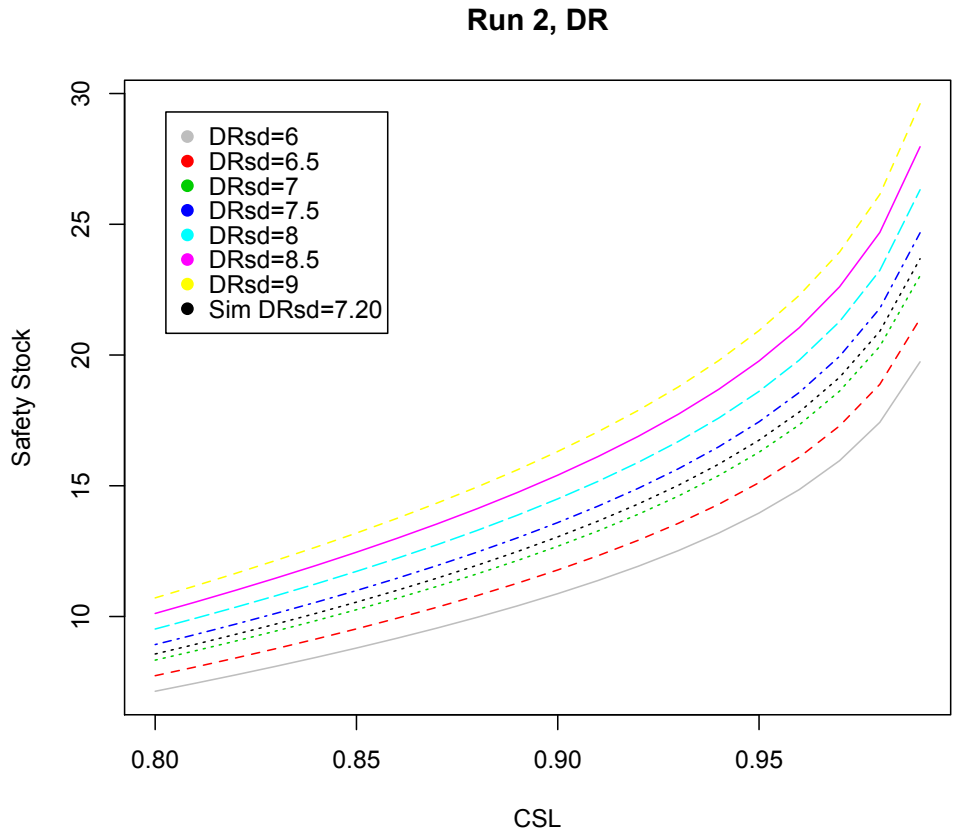


Figure 19: Plot of required safety stock by cycle service level for DR based on Run 2 simulated demand and assumed demand for a range of standard deviations

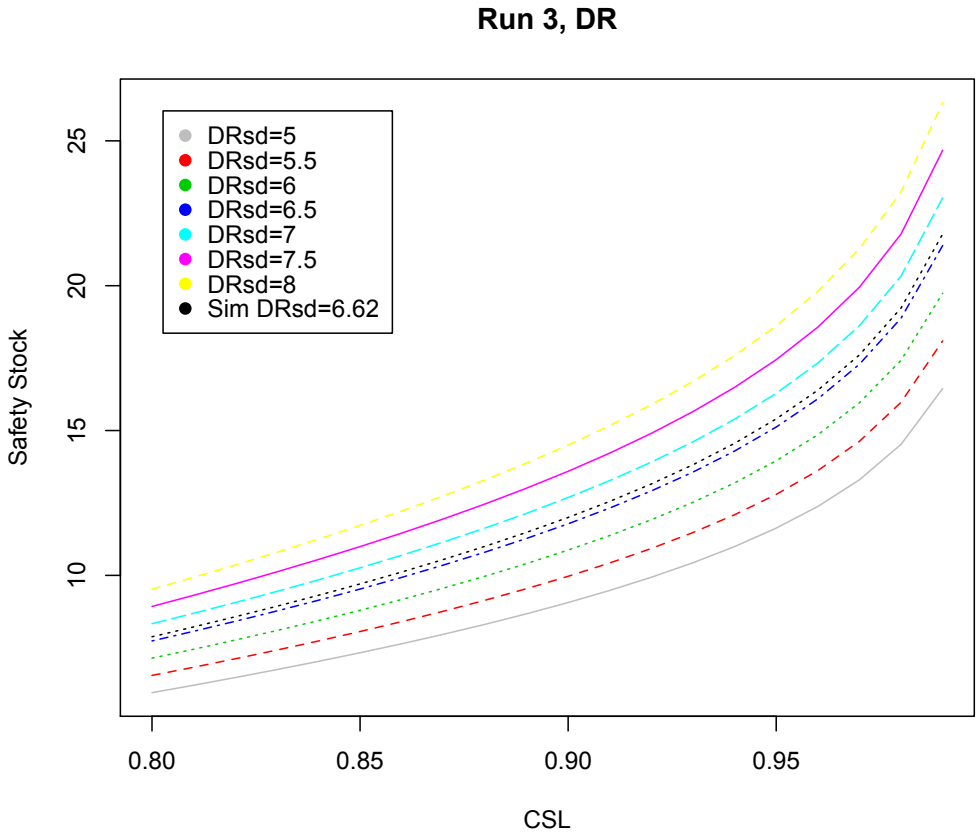


Figure 20: Plot of required safety stock by cycle service level for DR based on Run 3 simulated demand and assumed demand for a range of standard deviations

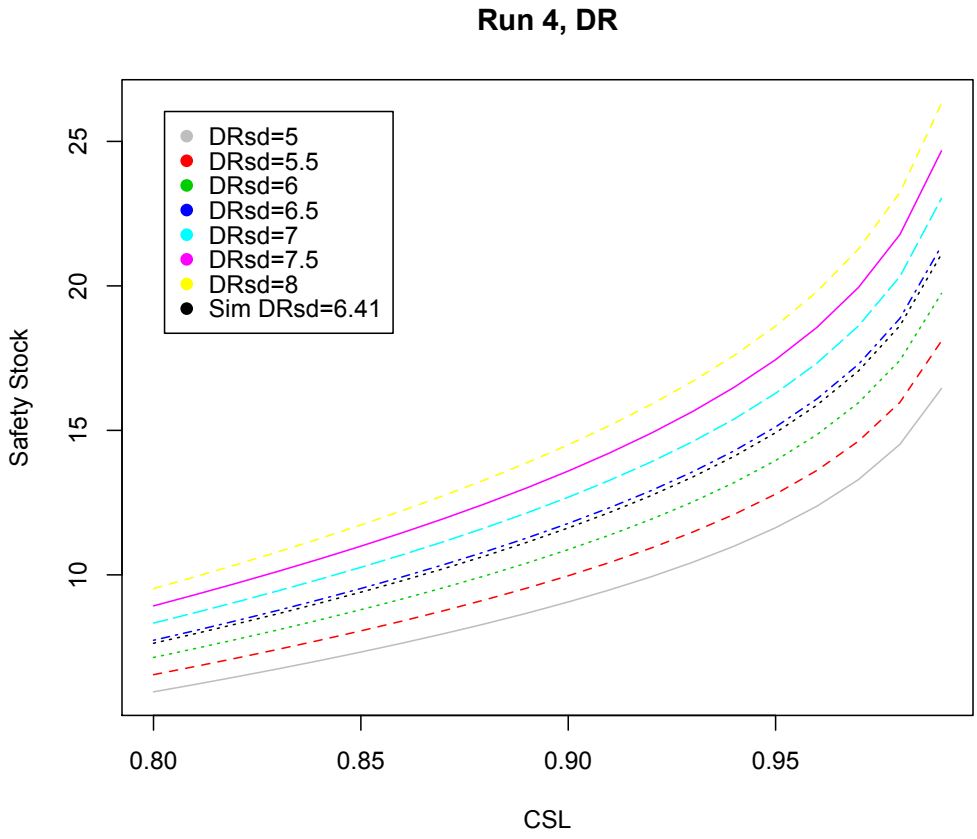


Figure 21: Plot of required safety stock by cycle service level for DR based on Run 4 simulated demand and assumed demand for a range of standard deviations

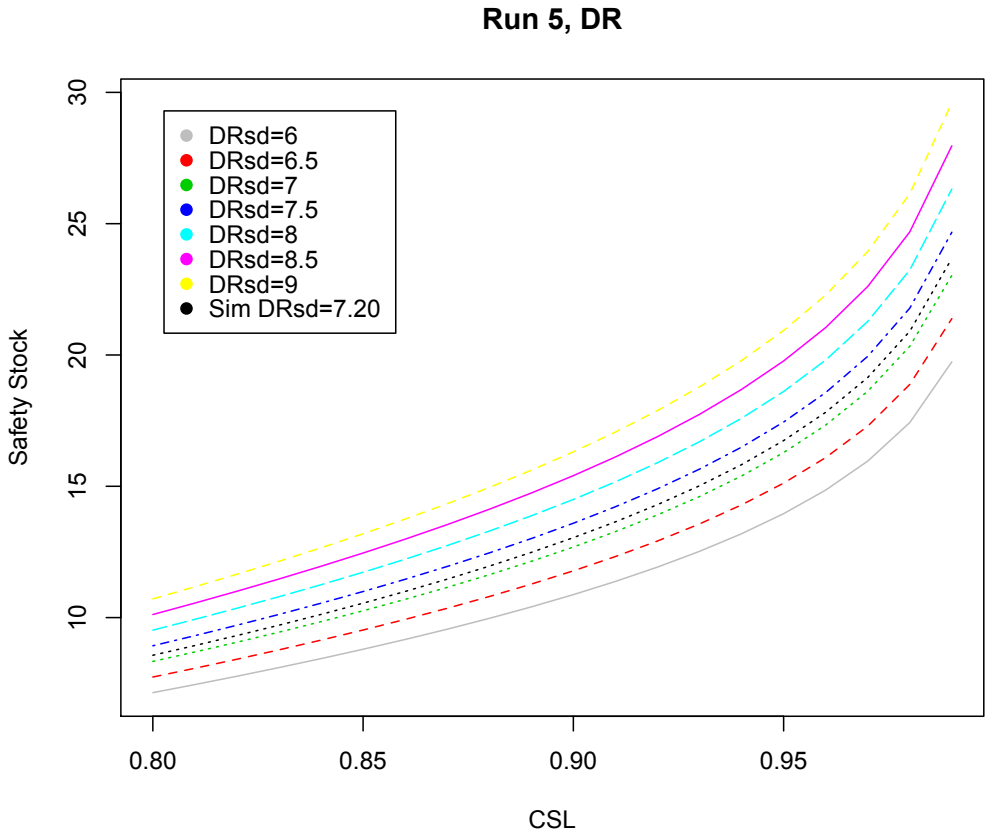


Figure 22: Plot of required safety stock by cycle service level for DR based on Run 5 simulated demand and assumed demand for a range of standard deviations

Furthermore, summary statistics are calculated for the predicted safety stock levels for simulated demand and standard deviation across all five runs for BU and DR. To do this, cycle service levels of 90%, 95%, and 99% are used. These statistics are summarized in Tables 3 and 4 below.

Table 3: Mean and standard deviation of predicted BU safety stock requirements across five simulation runs by cycle service level

BU Simulation Runs			
	90% CSL	95% CSL	99% CSL
Mean of SS Predicted	41.89886	53.7766	76.05727
SD of SS Predicted	2.064944	2.650327	3.748408

Table 4: Mean and standard deviation of predicted DR safety stock requirements across five simulation runs by cycle service level

DR Simulation Runs			
	90% CSL	95% CSL	99% CSL
Mean of SS Predicted	12.42034	15.94133	22.54613
SD of SS Predicted	0.6306886	0.8094801	1.144863

The standard deviations of the safety stock predictions across the five runs, for both BU and DR, are quite small. This means that for a specific cycle service level, safety stock predictions based on simulated demand and corresponding standard deviation of demand are fairly constant from run to run, especially for DR. For this problem, we only considered possible predicted standard deviations that are within about two units of the simulated standard deviations of demand. Realistically, forecasts may be off by much more than this, which means the safety stock requirements will be quite drastically under/over-estimated. This is especially significant when working with large or expensive inventory items.

4.3 Model Variations to Consider

The supply chain that we have used is a very simple two-stage supply chain with Uniformly distributed lead times. This assumption is not realistic in practice due to transportation delays and other unforseen issues. Because of this, the introduction of replenishment lead times with a more realistic distribution could be considered in future research and analysis. This would not be difficult to do, but it could provide even more insight into the performance of equation (1) because the σ_L term would be non-zero, meaning there would be two sources of variability to consider within the model.

Furthermore, an analysis taking the type of inventory into account could be considered. Introducing a holding cost or space constraint on the safety stock levels would add another level to the simulation model. Constraints like these are realistic, and it would allow us to quantify the significance of the results produced by the model.

Another possibility to consider is modeling a supply chain with more than two stages. This would allow us to analyze the safety stock equation by stage and see if the standard deviation of predicted safety stock levels increases as you move upstream. Based on these results, a conclusion could potentially be made regarding who (suppliers, distributors, retailers, etc.) takes on the most risk when using the safety stock equation.

4.4 Concluding Remarks

Section 1 introduced the topic and outlined some necessary background information about inventory, supply chains, and simulations. Section 2 was a discussion of

existing literature, formulas, and related probability distributions for the problem of multi-echelon inventory optimization. Section 3 then explained the formulation and implementation of a SimPy simulation model for analyzing safety stock levels within a supply chain. Finally, Section 4 has been a discussion of the results obtained from the simulation model and model variations that could be considered in future research. From these results, several conclusions can be drawn.

Based on Section 4.1, we can conclude that the simulation model and the theoretical model will provide similar results when normally distributed demand is present. This means if the demand you are working with is from a normal population, or the demand is an entire population that is normally distributed, the simulation model will not provide much insight beyond what the theoretical model gives. However, based on Section 4.2, we can conclude that when normally distributed demand is not present, the standard deviation of demand is a major factor in determining the performance of equation (1). If the forecasted standard deviation of demand is relatively accurate, even without normally distributed demand, the safety stock equation performs well. However, as the predicted standard deviation of demand becomes less accurate, the safety stock equation breaks down. Unless the inventory being analyzed consists of very small or inexpensive items, this result is significant. When working with large or expensive items, even a few units of safety stock can make a significant difference in storage space requirements or holding costs. Thus, companies who base their safety stock requirement levels solely on this safety stock equation run the risk of either holding too much safety stock or holding too little safety stock and having a stock out. Thus, it is difficult to quantify the significance of these conclusions because the

significance depends on the type of inventory being analyzed. However, we are able to conclude that a simulation model such as this one is useful in many situations. These situations include when demand is not assumed to be normally distributed, when the standard deviation of demand is not guaranteed to be extremely accurate, when replenishment lead times are not assumed to be normally distributed, and even when demand standard deviations are known to be accurate but a high cycle service level is desired.

BIBLIOGRAPHY

[1] K Balakrishnan. *Exponential distribution: theory, methods and applications*. CRC press, 1996.

[2] S. Chopra and P. Meindl. *Supply Chain Management: Strategy, Planning, and Operation*. Pearson Education, 2014.

[3] Martin Christopher. *Logistics & supply chain management*. Pearson Higher Ed, 2016.

[4] Stephen C Graves and Sean P Willems. Optimizing strategic safety stock placement in supply chains. *Manufacturing & Service Operations Management*, 2(1):68–83, 2000.

[5] RV Hogg, A Craig, and JW McKean. Intro. to mathematical statistics, 2005.

[6] Karl Inderfurth. Safety stock optimization in multi-stage inventory systems. *International Journal of Production Economics*, 24(1):103–113, 1991.

[7] Robert Jacobs and Richard Chase. *Operations and supply chain management*. McGraw-Hill Higher Education, 2013.

[8] Norman Matloff. A discrete-event simulation course based on the simpy language. *University of California at Davis*, 2006.

[9] Rick Pay. Avoiding obsolete inventory: Possession is 9/10ths of the problem. *As Published in IndustryWeek*, 2010.

[10] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

[11] Richard L Scheaffer and Linda Young. *Introduction to Probability and its Applications*. Cengage Learning, 2009.

[12] Chris Sheridan. *The Python Language Reference Manual*. Lulu.com, 2016.

[13] M Mahdi Tajbakhsh, Saeed Zolfaghari, and Chi-Guhn Lee. Supply uncertainty and diversification: a review. In *Trends in Supply Chain Design and Management*, pages 345–368. Springer, 2007.

[14] Jerzy Tyszer. *Object-oriented computer simulation of discrete-event systems*, volume 10. Springer Science & Business Media, 2012.

[15] T Vignaux and K Muller. Simpy: Documentation, 2015.

[16] Prem Vrat. Basic concepts in inventory management. In *Materials Management*, pages 21–36. Springer, 2014.

[17] Sean P Willems. How inventory optimization opens pathways to profitability. *Supply Chain Management Review*, 15(2), 2011.

[18] Joel D Wisner, Keah-Choon Tan, and G Keong Leong. *Principles of supply chain management: a balanced approach*. Cengage Learning, 2014.

APPENDICES

A SimPy Model

```
1 class P:
2     # External orders arrive to DR according to a Poisson process with
      daily mean of (150/10)/30=1/2 orders/day
3     externalToDRMean = 1/2
4     # External orders placed to DR have a lot size of 10 units/order
5     DRorderLotSize = 10
6     # External orders arrive to BU according to a Poisson process with
      daily mean of (50/20)/30=5/60 orders/day
7     externalToBUMean = 5/60
8     # External orders placed to BU have a lot size of 20 units/order
9     BUorderLotSize = 20
10    # BU places replenishment orders in lot sizes of 100 to upstream
      supplier
11    Q_1 = 100
12    # DR places replenishment orders in lot sizes of 20 to BU
13    Q_2 = 20
14    # Re-order point for BU is 20+(200/30)*7 units
15    ROP_BU = 20+(200/30)*7
16    # Re-order point for DR is 10+(150/30)*2 units
17    ROP_DR = 10+(150/30)*2
18    # Replenishment lead time from supplier to BU is 7 days
```

```
19      LT_1 = 7
20      # Replenishment lead time from BU to DR is 2 days
21      LT_2 = 2
22      # Run the simulation for 12 months
23      simulationTimeMax =  12 * 30
24
25  class S:
26      Inv = None
27      DRwaits = []
28      BUwaits = []
29      nBUCustomers = 0
30      nDRCustomers = 0
31      BU_Dem_day = list(repeat(0,P.simulationTimeMax))
32      DR_Dem_day = list(repeat(0,P.simulationTimeMax))
33
34  class Inventory:
35      def __init__(self, env):
36          self.env = env
37          self.BU_inv = simpy.Container(env, init = P.ROP_BU)
38          self.DR_inv = simpy.Container(env, init = P.ROP_DR)
39          self.mon_procBU = env.process(self.monitor_BU_inv(env))
40          self.mon_procDR = env.process(self.monitor_DR_inv(env))
41
42      def monitor_BU_inv(self, env):
43          while True:
```

```
44            if self.BU_inv.level <= P.ROP_BU:
45                print(''Time {0}: BU inventory reached ROP: BU places
    replenishment order".format(self.env.now))
46                yield self.env.timeout(P.LT_1)
47                print(''Time {0}: BU replenishment inventory arrives".
    format(self.env.now))
48                yield self.BU_inv.put(P.Q_1)
49                print(''Time {0}: BU replenishment order is added to
    inventory".format(self.env.now))
50            yield self.env.timeout(1)
51
52     def monitor_DR_inv(self, env):
53         while True:
54             if self.DR_inv.level <= P.ROP_DR:
55                 print(''Time {0}: DR inventory reached ROP: DR places
    replenishment order to BU".format(self.env.now))
56                 yield self.BU_inv.get(P.Q_2)
57                 print(''Time {0}: BU fills DR replenishment request".
    format(self.env.now))
58                 yield self.env.timeout(P.LT_2)
59                 print(''Time {0}: DR replenishment inventory arrives
    from BU".format(self.env.now))
60                 yield self.DR_inv.put(P.Q_2)
61                 print(''Time {0}: DR replenishment order is added to
    inventory".format(self.env.now))
```

```
62              yield self.env.timeout(1)
63
64 class DRCustomer(object):
65     def __init__(self, env, name = ''):
66         self.env = env
67         self.action = self.env.process( self.ordertoDR( ) )
68         if( name == ''):
69             self.name = 'RandomDRCustomer'+ str(randint(100))
70         else:
71             self.name = name
72
73     def DRorderToBU(self):
74         print(''Time {1}: DR places order to BU to fill order for {0}".
      format(self.name, self.env.now ) )
75         yield S.Inv.BU_inv.get(P.DRorderLotSize)
76         yield self.env.timeout(P.LT_2)
77         yield S.Inv.DR_inv.put(P.DRorderLotSize)
78
79     def ordertoDR(self):
80             startTime_DR = self.env.now
81             j = math.floor(self.env.now)
82             S.DR_Dem_day[j] += 1
83             print(''Time {1}: {0} places order to DR".format(self.name,
      self.env.now ) )
84             if S.Inv.DR_inv.level < P.DRorderLotSize:
```

```
85                  self.env.process(  self.DRorderToBU()    )
86              yield S.Inv.DR_inv.get(P.DRorderLotSize)
87              print(''Time {1}: {0} receives order from DR".format(self.
    name, self.env.now ) )
88              waitTime_DR = self.env.now - startTime_DR
89              print(''{0} had to wait {1} days".format( self.name,
    waitTime_DR  ))
90              S.DRwaits.append( waitTime_DR )
91
92 class BUCustomer(object):
93      def __init__(self, env, name = ''):
94          self.env = env
95          self.action = self.env.process( self.ordertoBU( ) )
96          if( name == ''):
97              self.name = 'RandomBUCustomer'+ str(randint(100))
98          else:
99              self.name = name
100
101      def ordertoBU(self):
102          startTime_BU = self.env.now
103          i = math.floor(self.env.now)
104          S.BU_Dem_day[i] += 1
105          print(''Time {1}: {0} places order to BU".format(self.name, self
    .env.now ) )
106          yield S.Inv.BU_inv.get(P.BUorderLotSize)
```

```
107         print(``Time {1}: {0} receives order".format(self.name, self.env
        .now ) )
108         waitTime_BU = self.env.now - startTime_BU
109         print(``{0} had to wait {1} days".format( self.name,
        waitTime_BU  ))
110         S.BUwaits.append( waitTime_BU )
111
112 class DROrderProcessor(object):
113     def __init__(self, env, DRlambda):
114         self.env = env
115         self.action = env.process(self.DREntrance() )
116         self.lam = DRlambda
117
118     def DREntrance(self):
119         while True:
120             interarrivalTime_DR = Exponential( scale = 1/P.
        externalToDRMean )
121             yield self.env.timeout( interarrivalTime_DR )
122             c = DRCustomer(self.env, name = ``DRCustomer {0}".format(S.
        nDRCustomers))
123             S.nDRCustomers += 1
124
125 class BUOrderProcessor(object):
126     def __init__(self, env, BUlambda):
127         self.env = env
```

```
128             self.action = env.process(self.BUEntrance() )
129             self.lam = BUlambda
130
131         def BUEntrance(self):
132             while True:
133                 interarrivalTime_BU = Exponential( scale = 1/P.
        externalToBUMean   )
134                 yield self.env.timeout( interarrivalTime_BU )
135                 c = BUCustomer(self.env, name = ''BUCustomer {0}".format(S.
        nBUCustomers))
136                 S.nBUCustomers += 1
137
138 def model(randomSeed = 123):
139     seed(randomSeed)
140     S.DRwaits = []
141     S.BUwaits = []
142     envr = simpy.Environment()
143     BU = BUOrderProcessor(envr, BUlambda = P.externalToBUMean)
144     DR = DROrderProcessor(envr, DRlambda = P.externalToDRMean)
145     S.Inv = Inventory(envr)
146     envr.run(until = P.simulationTimeMax)
147     return S.DRwaits, S.BUwaits, S.BU_Dem_day, S.DR_Dem_day, S.
        nBUCustomers, S.nDRCustomers
```

B Shapiro-Wilk Test Results for Simulated Demand

BU Simulation Runs, Mean Daily Demand = 50		
Run	Test Statistic	P-value
1	0.9905	0.02045
2	0.9939	0.1577
3	0.9956	0.4127
4	0.9905	0.02051
5	0.9934	0.1195

DR Simulation Runs, Mean Daily Demand = 50		
Run	Test Statistic	P-value
1	0.988	0.004575
2	0.9923	0.06026
3	0.9902	0.01685
4	0.9954	0.3637
5	0.9937	0.1401

BU Simulation Runs, Mean Daily Demand = 5/60		
Run	Test Statistic	P-value
1	0.3136	$< 2.2 \cdot 10^{-16}$
2	0.2854	$< 2.2 \cdot 10^{-16}$
3	0.2383	$< 2.2 \cdot 10^{-16}$
4	0.2492	$< 2.2 \cdot 10^{-16}$
5	0.331	$< 2.2 \cdot 10^{-16}$

DR Simulation Runs, Mean Daily Demand = 1/2		
Run	Test Statistic	P-value
1	0.72	$< 2.2 \cdot 10^{-16}$
2	0.6944	$< 2.2 \cdot 10^{-16}$
3	0.6751	$< 2.2 \cdot 10^{-16}$
4	0.6961	$< 2.2 \cdot 10^{-16}$
5	0.7028	$< 2.2 \cdot 10^{-16}$

VITA

LAUREN HOLDEN

Education:	B.S. Mathematical Sciences, Clemson University, Clemson, South Carolina 2014 M.S. Mathematical Sciences, East Tennessee State University, Johnson City, Tennessee 2017
Professional Experience:	Undergraduate Research, Dr. Irina Viktorova, Clemson University, *Analysis of Resonance Frequencies for the Problem of Induced Vibrations Along the Human Arm*, Clemson, South Carolina, 2013-2014 Graduate Assistant, East Tennessee State University and North Side Elementary School, Johnson City, Tennessee, 2015–2017 Operations Research Intern, Eastman Chemical Company, Kingsport, Tennessee, 2017–present
Relevant Skills:	Python, SimPy, Pyomo, GLPK MathProg, R, SAS, Minitab, SQL, Tableau, Excel/Word/PowerPoint